تمهيـــــــــــــــــــــــــــــد Preface

من المعلوم اليوم أن الحاسبات انتشرت انتشاراً واسعاً وكبيراً لدرجة أنها أصبحت في كل موقع وفي كل مكان ولا يمكن الاستغناء عنها بأي حال من الأحوال، وذلك لما تقوم به من أعمال كبيرة وعظيمة ولما تتمتع به من قدرة عالية على إجراء العمليات الحسابية وغيرها من العمليات في وقت قصير جداً. كما أنها تتميز بالقدرات العالية على معالجة الكم الهائل من البيانات حفظاً وترتيباً واسترجاعاً وبحثاً، وغيرها الكثير من العمليات.

ونظراً لما سبق أصبح لزاماً علينا – لكي نواكب هذا العصر وننهض بوطننا وشعبنا وأمتنا – أن نعرف الكثير عن هذه الحاسبات وكيف يمكن التعامل معها والاستفادة منها. ومن الوسائل التي تساعدنا على الاستفادة من هذه الحاسبات معرفة وإتقان إحدى لغات البرمجة المعروفة والمشهورة هذه الأيام. ومن هذه اللغات المشهورة والتي تستخدم على نطاق واسع لغة C++، ولغة Java، ولغة C++ والتي ستكون محوراً لدراستنا في هذا الفصل الدراسي.

حيث أننا سنتعرف على هذه اللغة وتراكيبها وقواعدها الصياغية وكيفية تطوير البرامج من خلالها. ولكن قبل أن نبدأ، سنقوم أولاً بتأسيس البنية التحتية التي تمكننا من التعامل مع اللغة بيسر وسهولة. وذلك من خلال البدء بدراسة نظرية عن ماهية جهاز الحاسوب وأهميته ومكوناته وآلية عمله. ثم بعد ذلك نتطرق لكيفية تحليل المشاكل قبل البدء بحلها عملياً عن طريق لغات البرمجة وذلك من خلال التعرف على مفهوم الخوارزميات والمخططات الانسيابية.

وعلى الطالب أن لا يغفل عن تقييد ملاحظاته أثناء المحاضرة، وذلك لأن هناك مفاهيم أو أمثلة قد لا تكون مدونة في هذه الملزمة، التي تعتبر مرجعاً مساعداً للطالب، ولكنه ليس كافياً.

كما أن على الطالب التركيز على أخذ المعلومة بقوة والإحاطة بها من جميع جوانبها والتطبيق العملي أولاً بأول للبرامج الحاسوبية التي يُلزم بها، وإجابة الأسئلة البحثية التي تجعل الطالب شريكاً للمدرس في حصوله على المعلومة. وذلك حتى يكون قد استفاد بأكبر قدر ممكن من المادة العملية المطروحة. كما أن عليه التواصل مع مدرس المادة كلما سنحت الفرصة للسؤال والاستفسار عن المشاكل أو الصعوبات التي تعيقه في أي مرحلة من مراحل تلقي المادة العلمية.

عبد الفتاح عبد الرب المشرقي ١٢ / ٢٠١١م

١.١. ما هو الحاســـــــــــــوب؟ What is Computer?

"الحاسوب" (Computer) هو عبارة عن جهاز له القدرة على إنجاز عمليات حسابية واتخاذ قرارات منطقية بسرعة تتجاوز ملايين بل مليارات المرات من سرعة الكائن البشري. حيث يوجد هذه الأيام العديد من الحواسيب الشخصية التي تنجز مليار عملية "جمع" في الثانية الواحدة. وهذا الرقم من العمليات لو أمضى الإنسان حياته كاملة في إنجازها لما أتمها على الوجه الأمثل. بل يوجد الآن "كمبيوترات عملاقة" (Supercomputers) أصبحت قادرة على إتمام مئات المليارات من عمليات "الجمع" في الثانية الواحدة. ويتم تطوير حواسيب تصل قدرتها على المعالجة إلى أبعد من هذا الحد.

ويمكن تعريف الحاسوب أيضاً بأنه جهاز إلكتروني رقمي يستقبل مجموعة بيانات من البيئة الخارجية ليقوم بمعالجتها وإخراجها في شكل معلومات مفيدة أو يقوم بتخرينها لحين الحاجة إليها.

ومن هنا يتضح أن الحاسوب قادر على أداء أربع مهام أساسية هي:

١. إدخال البيانات (ويتم ذلك عن طريق وسائط الإدخال).
٢. معالجة البيانات (ويتم ذلك عن طريق وحدة المعالجة المركزية).
٣. إخراج البيانات في شكل معلومات مفيدة (ويتم ذلك عن طريق وسائط الإخراج).
٤. خزن البيانات أو المعلومات لحين الحاجة إليها (عن طريق وسائط الخزن المختلفة - الذواكر).

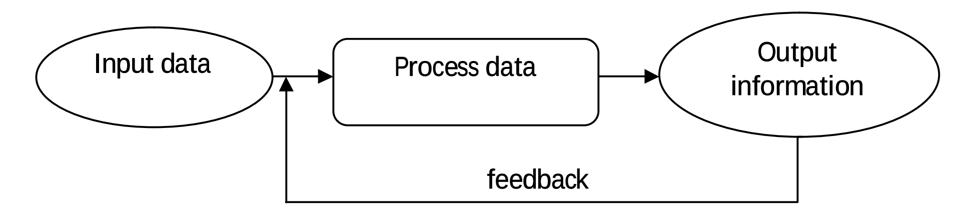
٢.١. البيانات والمعلومات Data and Information

"البيانات" (data) هي عبارة عن المادة الخام المدخلة إلى الحاسوب بغرض معالجتها للحصول منها على بيانات أخرى أكثر إفادة نسميها بـ "المعلومات" (information). فيمكننا القول أن المعلومات هي نتاج المعالجة الحاسوبية لمجموعة من البيانات المدخلة إلى الحاسوب.

وتتعدد أنواع البيانات التي يمكن إدخالها إلى الحاسوب. فهناك بيانات "رقمية" (numeric)، بيانات "حرفية" (characters)، بيانات "نصية" (text)، بيانات "صورية" (images)، بيانات "صوتية" (audio)، وبيانات "صوت وصورة" (video). وعلى هذا الأساس فقد تعددت وسائط الإدخال تبعاً لتعدد أنواع البيانات التي يمكن أن تدخل إلى جهاز الحاسوب (كما سنرى لاحقاً).

أما "المعلومات" (information)، فهي ناتج معالجة البيانات. و هناك عدة طرق يمكن بها إخراج المعلومات المفيدة الناتجة من عملية معالجة البيانات المدخلة على الحاسوب. وبالتالي فقد تعددت وسائط إخراج المعلومات، فيمكن إخراج المعلومات بشكل مرئي على "الشاشة" (screen) أو بشكل مطبوع عن طريق "الطابعة" (printer) أو بشكل مسموع عن طريق "السماعات" (speakers) .. الخ.

كما يجدر التنبيه إلى أن المعلومة يمكن أن تدخل مرةً أخرى على الحاسوب كبيانات جديدة بغرض الحصول منها على معلومات إضافية أو معلومات أكثر إفادة. وتسمى هذه العملية بـ "التغذية العكسية" (feedback).



٣.١. العتاديات والبرمجيات Hardware and Software

يتكون الحاسوب من أجهزة متعددة نطلق عليها "العتاديات" (hardware) (مثل لوحة المفاتيح، الشاشة، الأقراص، الذاكرة، الـ DVD ، الـ CD-ROM ووحدات المعالجة). أما البرامج التي تعمل على جهاز الحاسوب فنطلق عليها "البرمجيات" (Software). ولو قمنا بنظرة مماثلة بين الابتكار الحاسوبي والخلق البشري، لأمكننا القول بأن العتاديات تقابل أعضاء الجسد المحسوسة عن الإنسان، بينما تقابل البرمجيات الروح المسئولة عن نشاط الجسد وحركته ويقظته. وعليه، لا يمكن أن نتصور النظام الحاسوبي خالياً من المكونات العتادية الأساسية، وكذلك لا يمكننا تصوره خالياً من البرمجيات التي تبعث الحياة في الكتل المعدنية المختلفة التي أسميناها بالعتاديات.

١.٣.١. تنظيم الحاسوب (العتاديات) Computer Organization (Hardware)

بغض النظر عن الفروقات في المظهر الخارجي، فيمكن أن ننظر لجهاز الحاسوب على أنه مكون من أربعة أجزاء رئيسية:

١. "وحدة الإدخال" (Input unit). وهي عبارة عن الجزء المستقبل من جهاز الحاسوب. حيث تحصل على البيانات أو البرامج الحاسوبية عن طريق "أجهزة الإدخال" (input devices). الوسيلة الأساسية للإدخال هي "لوحة المفاتيح" (keyboard) ، وكذلك يمكن إدخال الأوامر إلى الحاسوب عن طريق جهاز التأشير المسمى بـ "الفأرة" (mouse). ويمكن إدخال المعلومات عن طريق التحدث إلى الحاسوب بـ "المايكروفون" (microphone) ، كذلك يمكن مسح الصور عن طريق "الماسح الضوئي" (scanner) ، أو التقاطها عن طريق "آلة التصوير الرقمية" (digital camera) أو عن طريق الشبكات (مثل شبكة الإنترنت) .. الخ.

٢. "وحدة الإخراج" (Output unit). وهي عبارة عن الجزء المرسل من الحاسوب. حيث تأخذ البيانات التي قام الحاسوب بمعالجتها ووضعها في "أجهزة إخراج" (output devices) لجعل المعلومات متاحة خارج الحاسوب. الوسيلة الأساسية للإخراج هي "الشاشة" (screen)، وقد يتم الإخراج عبر "الطابعة" (printer) أو "السماعات" (speakers) أو الشبكات (مثل شبكة الإنترنت).

٣. "وحدة الخزن" (Storage unit). وهي المستودع التي يتم فيه حفظ البيانات، كي نتمكن من الرجوع إليها وقت الحاجة إليها. وتتخذ أصنافاً وأشكالاً متعددة تختلف في أشكالها وأحجامها وسرعاتها. ويمكن تصنيفها تحت صنفين رئيسين هما:

a. "وحدة الذاكرة" (Memory unit). هي الجزء التخزيني القصير الأمد في الحاسوب. حيث تأخذ البيانات المدخلة عبر وحدات الإدخال بحيث تكون جاهزة للمعالجة في أي لحظة. أيضاً تقوم هذه الوحدة بأخذ المعلومات التي تم معالجتها كي ترسلها إلى أي وسيط إخراج. وتفقد المعلومات التي في هذه الوحدة بمجرد انقطاع التيار الكهربائي، ولهذا السبب تسمى "ذاكرة متطايرة" (volatile memory). ويطلق على هذه الوحدة أيضاً "الذاكرة" (memory) أو "الذاكرة الرئيسية" (primary memory).

b. "وحدة الخزن الثانوية" (Secondary storage unit). هي الجزء التخزيني الطويل الأمد في الحاسوب. ويتم فيها تخزين البرامج والبيانات بشكل دائم حتى يتم الرجوع إليها حين الحاجة إليها (مثل "القرص الصلب" hard drive). وهي بذلك لا تفقد بياناتها بمجرد انقطاع التيار الكهربائي، ولذلك تسمى "ذواكر غير متطايرة" (non-volatile memories). والوصول إلى بيانات هذه الوحدة يتطلب وقتاً أطول مما هو عليه الحال في وحدة الذاكرة الرئيسية، لكنها من حيث السعر أرخص. ومن الأمثل على هذه الوحدات الـ CDs وهي قادرة على تخزين مئات الملايين من الحروف. والـ DVDs وهي قادرة على تخزين مليارات من الحروف.

c. "وحدة المعالجة المركزية" (Central processing unit (CPU). وهي الجزء الإداري في الحاسوب. فهي تنسق وتشرف على عمليات الوحدات الحاسوبية المختلفة. فهي توجه وحدات الإدخال للقيام بإدخال البيانات إلى وحدة الذاكرة عند اللزوم. كذلك تخبر وحدة الحساب والمنطق متى ينبغي عليها أخذ المعلومة من وحدة الذاكرة لإتمام العلميات الحسابية والمنطقية. كما أنها تخبر وحدات الإخراج متى ينبغي عليها إرسال المعلومات من وحدة الذاكرة إلى أجهزة الإخراج. يوجد هذه الأيام نوع من المعالجات يطلق عليها "المعالجات المتعددة" (multiprocessors) وذلك لأنها قادرة على إتمام العديد من عمليات المعالجة بشكل متزامن. وهي تتكون من جزئين أساسيين هما:

- "وحدة الحساب والمنطق" (Arithmetic and logic unit (ALU). وهي الوحدة الإنتاجية في الحاسوب، حيث أنها مسئولة عن إتمام العمليات الحسابية مثل الجمع والطرح والضرب والقسمة. وتحتوي هذه الوحدة على آلية قادرة على اتخاذ القرار (أي تنفيذ العمليات المنطقية)، مثلاً لتحديد ما إذا كان عنصرين موجودين في وحدة الذاكرة متساويين أم لا.
- "وحدة التحكم" (.Control Unit (C.U. وهي الوحدة المسئولة عن تنسيق تبادل البيانات والتعليمات بين الذاكرة الرئيسية ووحدة الحساب والمنطق.

٢.٣.١. برمجيات الحاسوب Computer Software

يمكن وضع جميع برمجيات الحاسوب ضمن واحدة من التصنيفات الرئيسية التالية:

١. "نظم التشغيل" (.Operating Systems O.S).
٢. "البرامج التطبيقية" (Application Programs).
٣. "لغات البرمجة" (Programming Languages).

١.٢.٣.١. أولاً : "نظم التشغيل" (.Operating Systems O.S):

يعتبر "نظام التشغيل" (Operating System) أهم نوع من أنواع البرمجيات الحاسوبية. فهو يقوم بأداء مهمتين أساسيتين. الأولى، أنه يزود بـ "واجهة مستخدم" (user interface) تسمح للمستخدم من التفاعل بسهولة ويسر مع جهاز الحاسوب. والثانية، أنه يدير كافة الموارد الحاسوبية مثل وحدة المعالجة المركزية والذواكر وأجهزة الإدخال والإخراج وغيرها. فهو يحدد متى يُسمح بتنفيذ البرامج، ومتى يتم تحميلها في الذاكرة، وكيفية اتصال عتاديات الحاسوب مع بعضها البعض. فمهمة نظام التشغيل إذاً جعل التعامل مع الحاسوب سهلاً وضمان تشغيله بكفاءة وفاعلية.

وهناك اليوم العديد من أنظمة التشغيل المشهورة. ومنها Windows 2000 و Windows XP وهما إصداران من نظام التشغيل الذي تطوره شركة البرمجيات العملاقة Microsoft. ومن الأمثلة أيضاً نظام التشغيل المسمى Unix وهو أقل شعبية، وقد طورت منه إصدارة سميت Linux صارت أكثر شعبية. أما شركة Apple للحواسيب فقد طورت نظام التشغيل المسمى Mac OS والمعد خصيصاً للأجهزة المطورة من قبل هذه الشركة. والعديد العديد من أنظمة التشغيل المتوفرة والتي هي قيد الإنشاء والتطوير.

١.٣.٢.٢. ثانياً : "البرامج التطبيقية" (Application Programs):

"التطبيق" (application) هو عبارة عن مصطلح عام لأي برمجية غير نظم التشغيل. وهذه التطبيقات هي أكثر البرمجيات انتشاراً وتعدداً لأنها جاءت لحل مشاكل الحياة في جوانبها المختلفة وتسهيل التعامل معها. فهناك تطبيقات "معالجة الكلمة" (word processors) التي تهتم بالتعامل مع النصوص واجراء العمليات عليها مثل تغيير حجم النص، تغيير خط النص، ألوان النصوص، نسخ وقص ولصق النصوص، البحث عن نص معين .. الخ. ومن أشهر معالجات النصوص الموجودة هذه الأيام M.S Word المقدم من شركة البرمجيات العملاقة Microsoft. كما أن هناك تطبيقات لـ "الأوراق الانتشارية" (Spreadsheets) التي تستخدم في المعالجات الحسابية والمالية مثل البرنامج الشهير M.S. Excel. أيضاً توجد تطبيقات لـ "معالجة الصور" (image processing) مثل Paint و Photoshop. و "مديرات قواعد البيانات" (database managers) مثل M.S. Access. و "متصفحات الانترنت" (web browsers) مثل Internet Explorer. وتطبيقات أخرى للملفات الصوتية والفيديو والبريد الإلكتروني والألعاب وأنظمة التحكم وأنظمة المعلومات .. الخ. وتتميز التطبيقات البرمجية بأن لها "واجهات مستخدم" (user interface) خاصة تمكن المستخدم من التفاعل معها بشكل جيد.

١.٣.٢.٣. ثالثاً : "لغات البرمجة" (Programming Languages):

"**البرامج الحاسوبية**" Computer Programs

تستخدم الحواسيب في معالجة "البيانات" (data) مجموعة من التعليمات نطلق عليها "برامج حاسوبية" (Computer Programs) وهذه البرامج توجه الحاسوب في عمله من خلال مجموعة من الخطوات والأحداث المرتبة التي قام بكتابتها ما نطلق عليهم "مبرمجي الحاسوب" (computer programmers).

١.٣.٢.٤. المجالات البرمجية (Programming Domains):

دخلت الحواسيب في عدد ضخم من المجالات المختلفة، ابتداء من منشآت الطاقة النووية وصولاً إلى خزن السجلات والمعلومات الشخصية. وبسبب هذا التنوع الكبير في استخدام الحاسوب، فقد تم تطوير العديد من لغات البرمجة والتي لها أغراض متعددة. وهنا سنناقش بعضاً من تطبيقات الحاسوب البرمجية واللغات البرمجية المتصلة بها.

١.٣.٢.٤.١. التطبيقات العلمية (Scientific Applications)

تتطلب التطبيقات العلمية بنيات بيانية بسيطة ولكنها تتطلب عدداً كبيراً من العمليات الحسابية الحقيقية. البنيات الأكثر شيوعاً هي: "المصفوفات" arrays، وبنيات التحكم الأكثر شيوعاً هي "التكرار" looping و "الاختيارات" selections. وفي هذه التطبيقات غالباً ما تكون "الفاعلية" efficiency هي الأهم.

أمثلة: لغة FORTRAN **ولغة** ALGOL 60.

١.٣.٢.٤.٢. التطبيقات التجارية (Business Applications)

بدأ استخدام الحواسيب في التطبيقات التجارية في خمسينيات القرن الماضي.

أمثلة: لغة COBOL، **وهي أول لغة تجارية عالية المستوى ناجحة وقد ظهرت في الستينات وما تزال اللغة الأكثر استخداماً في هذا المجال. ومن خصائصها ما يلي:**

١. **وجود تسهيلات لتوليد تقارير مطورة.**
٢. **طرق دقيقة لوصف وخزن الأرقام العشرية والبيانات الحرفية.**
٣. **القدرة على تحديد عمليات رياضية عشرية.**

بالإضافة إلى لغات البرمجة ظهرت أدوات برمجية خاصة تستخدم في الحواسيب الصغيرة في المجالات التجارية، مثل "أنظمة الأوراق الانتشارية" spreadsheets systems **و "أنظمة قواعد البيانات"** database systems.

١.٣.٢.٤.٣. الذكاء الاصطناعي (Artificial Intelligence):

تمتاز تطبيقات الذكاء الاصطناعي بما يلي:

١. **استخدام الحسابات "الرمزية"** symbolic **بدلاً من "الرقمية"** numeric: **وفي العمليات الحسابية الرمزية يتم معالجة رموز مكونة من أسماء وليس من أرقام.**
٢. **العمليات الحسابية الرمزية تتم بشكل أكفأ عن طريق استخدام "القوائم المتصلة"** linked lists **وليس "المصفوفات"** arrays.
٣. **يتطلب هذا النوع من البرمجة أحياناً مرونة أكبر من المجالات البرمجية الأخرى.**

أمثلة:

١. **لغة** LISP: **وهي "لغة وظيفية"** functional language، **ظهرت هذه اللغة في عام ١٩٥٩م، وبهذه اللغة كتبت أغلب تطبيقات الذكاء الاصطناعي.**
٢. **لغة** Prolog: **وهي "لغة منطقية"** logic language، **ظهرت في بداية السبعينات.**

١.٣.٢.٤.٤. برمجة الأنظمة (Systems Programming)

"برمجيات النظم" systems software: **هي عبارة عن "نظام التشغيل"** operating system **وكل أدوات الدعم البرمجي في النظام الحاسوب. ولها الخصائص التالية:**

١. **في الأغلب تستخدم باستمرار.**
٢. **يجب أن تكون كفاءتها التنفيذية محسنة.**

مثال: "نظام تشغيل يونكس" Unix O.S.، **مكتوب بلغة** C **مما جعله نظام قابلاً للحمل في الآلات المختلفة، وذلك لأن لغة** C **تمتاز بالخصائص التالية:**

١. **مستوى قريب من المتدني.**
٢. **كفاءة تنفيذية عالية.**   ٣. **لا ترهق المستخدم بقيود الحماية الزائدة.**

١.٣.٢.٤.٥. اللغات النصية (Scripting Languages)

تستخدم مثل هذه اللغات عن طريق وضع مجموعة من الأوامر في ملف ثم تنفيذها.

أمثلة: لغة sh أو shell: وهي أول هذه اللغات، وقد بدأت كمجموعة صغيرة من الأوامر التي تفسر بعد ذلك كنداءات للبرامج الفرعية للنظام والتي تؤدي وظائف مفيدة مثل إدارة الملفات وتنقيتها.

١.٣.٢.٤.٦. لغات الأغراض الخاصة (Special-Purpose Langusges)

ظهرت هذه اللغات منذ أكثر من ٤٠ سنة. وهي تخدم مجالات متعددة من مجالات الحياة، ومنها ما يلي:

١. **لغات الـ RPG: وتستخدم لتوليد تقارير تجارية.**
٢. **لغات الـ APT: لتوليد أدوات ميكانيكية قابلة للبرمجة.**
٣. **لغات GPSS: وتستخدم في أنظمة المحاكاة.**

١.٣.٢.٥. بعض لغات المستوى العالي

من أكثر اللغات عالية المستوى استخداماً لغات C، ++C، لغات NET. من مايكروسوفت (مثل: Visual Basic.NET، Visual C++.NET و #C) ولغة Java. كذلك يمكن اعتبار لغة ++C على أنها واحدة من أكثر اللغات شهرة في مجال التعليم والتدريب البرمجي.

تاريخ لغتي C و ++C

تطورت لغة ++C والتي طورت من لغة C، والتي بدورها تطورت من لغتي BCPL و B. ولغة BCPL طورت عام ١٩٦٧م من قبل "مارتن ريتشارد" (Martin Richard) كلغة لكتابة برمجيات أنظمة التشغيل والمترجمات. بعدها جاء "كن ثومبسون" (Ken Thompson) والذي أدخل مزايا عديدة على لغة B الخاصة به واستخدمها في برمجة الإصدارات القديمة من نظام التشغيل "يونكس" (UNIX) في معامل شركة "بل" (Bell) عام ١٩٧٠م.

تطورت لغة C من لغة B من قبل "دينيس ريتشي" (Dennis Ritchie) في معامل شركة بل عام ١٩٧٢م. وأصبحت معروفة بأنها لغة تطوير نظام التشغيل يونكس. أما اليوم فإن أغلب شفرات نظم التشغيل المتعددة الأغراض (مثل تلك الموجودة على الأجهزة المحمولة والمكتبية ومحطات العمل والمزودات الصغيرة) مكتوبة بلغتي C و ++C.

لغة ++C والتي تعتبر امتداداً للغة C، طورت من قبل "بارني ستروستراب" (Bjarne Stroustrup) في بدايات ثمانينات القرن الماضي في معامل شركة بل. وجاءت ++C بمزايا جديدة ومتعددة زادت من أناقة لغة C. ولعل أهم الميزات التي دخلت على لغة ++C أنها تبنت منهجية "البرمجية الموجهة بالكائنات" (Object Oriented Programming) إلى جانب منهجية C "الهيكلية" (Structured Programming). ولكن تعتبر جافا هي أكثر اللغات في العالم التي تعتمد أسلوب البرمجة الموجه بالكائنات.

تاريخ الجافا Java

أثرت المعالجات الدقيقة تأثيراً عميقاً في الأجهزة الإلكترونية الاستهلاكية الذكية. ولنعرف كيف هذا، نرى أن شركة "صن مايكروسيستمز" (Sun Microsystems) قامت في عام ١٩٩١م بتمويل مشروع بحثي داخلي سمي "جرين" (Green) والذي أدى إلى تطور لغة معتمدة على لغة C++ كان مبتكرها هو "جيمس جوسلينج" (Games Gosling) وسميت حينذاك لغة "أوك" (Oak) أو – البلوط – نسبة إلى شجرة البلوط التي كانت موجودة خارج نافذة جيمس في شركة صن. ولكن بعد ذلك اكتشف أن هناك لغة أخرى كانت قد سميت بهذا الاسم. وبينما قامت مجموعة من شركة صن بزيارة مقهى محلي، ظهر للواجهة اسم جافا، ومن حينها التصق هذا الاسم بهذه اللغة.

واجه مشروع جرين بعض الصعوبات. فلم تكن سوق الأجهزة الإلكترونية المستهلكة الذكية قد تطورت في بداية تسعينيات القرن الماضي. وهذا أنذر بخطر إلغاء هذا المشروع. ولكن لحسن الحظ، فقد انطلقت ثورة الشبكة العالمية الموسعة في عام ١٩٩٣م ، وهذا جعل شركة صن تفكر تفكر في استخدام جافا لإضافة "محتوى حركي" (dynamic content) على صفحات الوب، مثل التفاعليات والرسوم المتحركة. وهذا بالطبع كتب عمراً جديداً للمشروع.

أعلنت شركة صن بشكل رسمي عن لغة جافا في مؤتمر كبير في مايو عام ١٩٩٥م. اكتسبت جافا من لحظتها اهتماماً وتأييداً واسعين في مجتمع المال والأعمال بسبب ظاهرة الاهتمام الكبيرة بالشبكة العالمية الموسعة. تستخدم جافا الآن في تطوير تطبيقات مشاريع ضخمة بغرض تحسين الأداء لـ "مزودات الوب" (web servers) (وهي تلك الحواسيب التي تزود بالمحتوى الذي نراه على "مستعرضات الوب" (web browsers)). وكذلك تعمل تطبيقات للأجهزة الاستهلاكية (مثل : الهواتف الخلوية والبيجرات والـ PDAs). ولها استخدامات أخرى متعددة.

لغة فورتران FORTRAN

طورت "لغة فورتران" FORTRAN (FORmula TRANslator) من قبل شركة IBM في منتصف خمسينيات القرن الماضي للتطبيقات العلمية والهندسية التي تتطلب عمليات حسابية معقدة. وهي ما زالت مستخدمة بشكل واسع خصوصاً في التطبيقات الهندسية.

لغة كوبول COBOL

أما لغة "كوبول" COBOL (COmmon Business Oriented Language) فطورت في أواخر خمسينيات القرن الماضي بواسطة مصنعي حواسيب والحكومة الأمريكية ومستخدمي الحواسيب الصناعية. وتستخدم كوبول للتطبيقات التجارية التي تتطلب معالجة دقيقة وذات كفاءة عالية لكميات كبيرة من البيانات. وما زال الكثير من البرمجيات التجارية مبرمج بلغة كوبول.

لغة باسكال Pascal

خلال ستينات القرن الماضي، بدأ الناس يدركون أن تطوير البرمجيات كان نشاطاً معقداً للغاية بشكل لم يكونوا يتخيلوه. وهذا حدا بهم إلى البحث عن منهجيات برمجية أكثر فاعلية وكفاءة، حتى توصل الباحثون في نفس العقد من القرن الماضي ابتكار "البرمجة الهيكلية" (structured programming)، وهو أسلوب أوضح وأسهل في تعديل البرامج واختبارها وتصحيحها. ومن أوائل اللغات التي تبنت هذه المنهجية لغة "باسكال" (Pascal) المطورة من قبل البروفيسور "نيكلاوس ويرث" (Nicklaus Wirth) في عام ١٩٧١م . وقد سميت هذه اللغة بهذا الاسم نسبة إلى الفيلسوف والرياضي المشهور "بليز باسكال" (Blaise Pascal) الذي عاش في القرن السابع عشر الميلادي. وصممت بغرض تدريس البرمجة الهيكلية في البيئات الأكاديمية. وأصبحت بسرعة لغة

البرمجة المفضلة في أغلب الكليات. وتفتقر لغة باسكال إلى العديد من المزايا التي تجعلها مفيد في التطبيقات التجارية والصناعية والحكومية، لذلك فهي لم تكن مقبولة بشكل واسع في هذه البيئات.

لغة إيدا Ada

صممت لغة "إيدا" (Ada) **تحت رعاية وزارة الدفاع الأمريكية في سبعينيات وثمانينيات القرن الماضي. وقد صممتها لتلبية أغلب احتياجاتها. وقد سميت هذه اللغة بهذا الاسم نسبة إلى "ليدي إيدا لفليس"** (Lady Ada Lovelace) **ابنة الأديب "لورد بايرن"** (Lord Byron)**. وهذه السيدة كانت أول من كتب برنامج حاسوبي في العالم في بدايات القرن التاسع عشر، وذلك لـ "جهاز الآلة التحليلية للحوسبة الميكانيكية"** (Analytical Engine mechanical computing device) **المصمم من قبل العالم "تشارلس بابج"** (Charles Babbage)**. ومن أهم ميزات لغة** Ada **قدراتها على أداء العديد من المهام على التوازي (في نفس الوقت)، وهذا ما يطلق عليه "تعدد المهام"** (multitasking)**. أما لغة جافا فلها نفس المقدرة من خلال ما يسمى بـ** "multithreading".

بيسك، فيجوال بيسك، فيجوال سي++، سي شارب و دوت نت

لغة "BASIC" **طورت في منتصف ستينات القرن الماضي في "كلية دارت ماوث"** (Dartmouth college)**. وهي اختصار لـ** (Beginner's All-Purpose Symbolic Instruction Code) **أي "شفرة التعليمات الرمزية لأغراض العامة للمبتدئين". وهي مصممة كوسيلة لكتابة البرامج البسيطة.**

لغة "فيجوال بيسك" (Visual Basic) **طورت في بداية تسعينيات القرن الماضي لتبسيط تطوير تطبيقات النوافذ المقدمة من شركة مايكروسوفت. وقد أصبحت واحدة من أشهر اللغات شعبية في العالم. ومن آخر أدوات التطوير التي ابتكرتها مايكروسوفت ما يسمى بـ "بيئة دوت نت"** (.NET platform) **وهي جزء من استراتيجيات مايكروسوفت الرامية إلى دمج الإنترنت والوب في التطبيقات الحاسوبية. وهذه البيئة تزود المطورين بالمقدرات التي يحتاجونها لإنشاء وتشغيل التطبيقات الحاسوبية التي يمكن تشغيلها على أجهزة الحاسوب الموزعة عبر شبكة الانترنت. وأهم ثلاث لغات رئيسية ابتكرتها مايكروسوفت هي "فيجوال بيسك دوت نت"** (Visual Basic .NET) **، وهي لغة مبنية على أساس لغة بيسك. و "فيجوال سي++ دوت نت"** (Visual C++ .NET) **المبنية على أساس لغة سي++. ومن ثم لغة "شي شارب"** (C#) **المطورة من لغتي سي++ وجافا لتعمل على بيئة دوت نت. يمكن للمستخدمين الذين يستخدمون بيئة دوت نت البرمجية كتابة المكونات البرمجية باللغة البرمجية التي اعتادوا عليها، ومن ثم بإمكانهم تركيب هذه المكونات البرمجية مع أي مكونات أخرى مكتوبة بلغات دوت نت.**

١.٢. مقدمـــــــــــــــــــة Introduction

في هذا الفصل سنعرض مقدمة عن ماهية برنامج الحاسوب، وأهمية مهنة البرمجة. ثم بعد ذلك نشرح القواعد التي تساعد في تحليل المشكلة ومعرفة عناصرها المكونة لها، وكيف يمكن تجزئة المشكلة إلى أجزاء صغيرة يسهل التعامل معها. وفيها أيضاً نوضح رموز رسم خرائط التدفق، ثم رسم هذه الخرائط للمشكلة بعد كتابة الخوارزمية، والتي تعطي صورة لحل المشكلة.

٢.٢. البرنامج الحاسوبي Computer Program

"البرنامج الحاسوبي" (Computer Program): هو عبارة عن مجموعة من التعليمات المتسلسلة والمكتوبة بلغة برمجة معينة، يمكن استخدامه لتوجيه الحاسوب بشكل مباشر لإتمام بعض المهام الحسابية.

والبرنامج هو الذي يحدد للحاسوب كيفية التعامل مع البيانات للحصول على النتائج المطلوبة. ويكتب من قِبَل شخص يسمى "مبرمج الحاسوب" (Computer Programmer). وهذا الشخص يقوم أولاً بفهم المشكلة ويقترح الحل وينفذه لحل هذه المشكلة. ويجب أن يكون البرنامج في مجموعِهِ واضحاً وصحيحاً وليس فيه لبس أو غموض. وأنت كطالب برمجة منوط بك أن تكون مبرمجاً ناجحاً. وسنتكلم بعد قليل عن أهمية مهنة المبرمج.

٣.٢. البرمجة : ماهيتها وأهميتها Programming: Essence and Importance

عندما نقوم بكتابة برنامج بواسطة الحاسوب الآلي، فإن معنى ذلك أننا نقوم بإعطائه التعليمات والأوامر اللازمة لتنفيذ تعليمات معينة. ومن هنا، فقد عرفنا برنامج الحاسوب سابقاً وقلنا بأنه : مجموعة من الأوامر والتعليمات التي تعطى للحاسوب الآلي للقيام بأعمال مرتبة ومحددة.

وكما أن الواحد منا لا يستطيع القيام بتعليمات المدرس إلا بعد أن يتلقى تلك التعليمات بلغة يستطيع فهمها كاللغة العربية أو اللغة الإنجليزية مثلاً، فإن الحاسوب كذلك لا يستطيع تلقي تلك التعليمات والأوامر إلا بعد أن تكون مكتوبة بإحدى اللغات التي يستطيع الحاسوب فهمها والتعامل معها. وكل لغة من هذه اللغات لها أوامرها وتعليماتها الخاصة بها، ولكن جميعها تتفق بأن الحاسوب يقوم بعملٍ ما جراء هذه التعليمات.

٤.٢. أهمية مهنة المبرمج Programmer Job Importance

من المعلوم أن الذي يقوم بكتابة البرامج لحل المشكلات الكثيرة والمعقدة هم المبرمجون. ولا يمكن الاستغناء عنهم بأي حال من الأحوال، لأن دورهم مهم وحيوي، وتكثر الحاجة لهم في شتى المجالات، وذلك لعمل الآتي:

١- كتابة البرامج وبناء الأنظمة المختلفة لحل المشاكل وتبسيط التعامل مع الحاسوب.
٢- المسئولية الكاملة عن إصلاح ما يحدث من أعطال أو حل المشاكل التي تحدث في الأنظمة المختلفة.
٣- بناء واجهة المستخدم المختلفة في كثير من اللغات والتطبيقات.
٤- بناء نظم التشغيل المختلفة مثل Unix، و Windows، وغيرها من النظم. فمثلاً استخدمت لغة C في بناء نظام التشغيل Unix.

٥.٢. صناعة البرمجيات Software Manufacturing

تعتبر صناعة البرمجيات في عصرنا الحالي من الصناعات المهمة جداً، والتي تتطور باستمرار نتيجة التطور الهائل في صناعة الحواسيب الآلية. ولذلك فإن هذه الصناعة تتطلب مبرمجين مهرة ولديهم القدرة على تحليل وحل المشاكل، بالإضافة إلى الإلمام بكل المستجدات والعلوم والتطوير المتعلق بالحاسوب وصناعة الحواسيب، وذلك حتى يستطيعوا مواكبة تطوير البرامج والنظم المختلفة للاستفادة من التقدم في تكنولوجيا الحواسيب.

٦.٢. تطوير البرمجيات Software Development

عندما تواجهنا مشاكل في حياتنا اليومية يتطلب منا حلها بأن نقوم بأعمال متكررة أو عمليات حسابية معقدة أو عمليات محددة مملة، فعندئذٍ نحتاج إلى كتابة برامج لمساعدتنا لإنجاز حل هذه المشاكل بسهولة ويسر.

إن الحاسوب لا يستطيع حل جميع المسائل والمشاكل التي تواجهنا في حياتنا اليومية وإن كانت بسيطة في نظرنا، كتحديد الوجبة المفضلة لديك أو لدى زميلك، أو القيام باختيار الكلية التي تحقق طموحاتك. فمثل هذه المشاكل يستحيل على الحاسوب أن يقوم بحلها. ومن ناحية أخرى فإن الحاسوب سيطير فرحاً – إن صح التعبير – عندما نكلفه بحل مشاكل فيها عمليات حسابية متعلقة بالأرقام ومعالجتها، فهذا يعتبر مجالاً شيقاً بالنسبة للحاسوب لا يجاريه في ذلك أي جهاز أو آلة أخرى.

١.٦.٢. مراحل تطوير البرامج Programs Development Phases

إذا أردت بناء منزل، أو عمل مشروع معين فلا بد من دراسة المشروع وتحليله تحليلاً دقيقاً حتى تتمكن من تحقيق التصميم المناسب وتستخدم في ذلك الأدوات المناسبة. بعدها تبدأ بالتنفيذ خطوة بخطوة، ومن ثم تقيم الخطوات التي قمت بها وتصحح الأخطاء إن وجدت إلى أن تنتهي من المشروع والذي يحتاج بعدها إلى صيانة دورية للحفاظ على جودته أو تطوير ميزاته. بمثل هذا الأسلوب يتم بناء وتطوير البرامج. ولحل المشاكل بواسطة الحاسوب لتكتمل في النهاية على شكل برنامج يستطيع الحاسوب فهمه والتعامل معه، فإن هناك خطوات ومراحل لازمة يمر بها المبرمج في حل تلك المشاكل، وهي :

١.١.٦.٢. "أولاً : تحديد وتعريف المشكلة" (Problem Definition):

في هذه الخطوة يقوم المبرمج بتحديد وتعريف المشكلة، وتتضمن هذه الخطوة تحديد التالي بالترتيب:

١- الهدف من البرنامج: كأن يكون – على سبيل المثال – لحساب صافي الأرباح، أو الرواتب، أو فواتير استهلاك الكهرباء والماء، أو حساب المعدل التراكمي للطلاب.
٢- نوع وحجم البيانات المدخلة ووسائل الإدخال: والمدخلات هي البيانات اللازم الحصول عليها لمعرفة النتائج والمخرجات.
٣- نوع وحجم المخرجات ووسائل الإخراج (تقارير – فواتير – شيكات – نقود ....): والمخرجات هي النتائج أو المعلومات المراد التوصل إليها من حل المشكلة.
٤- تحديد عمليات المعالجة: ونعني بها تحديد العمليات الحسابية والخطوات المنطقية التي نقوم بإجرائها على مدخلات البرنامج حتى تؤدي في النهاية إلى المخرجات والنتائج السليمة.
٥- مستخدمي البرنامج والمستفيدين منه.

مثال: نفترض أننا نريد حساب مساحة المستطيل بمعلومية الطول والعرض، قم بتحليل عناصر المشكلة إذا علمت أن : مساحة المستطيل = الطول × العرض.

الحل:

١- **نحتاج إلى إدخال الطول (L) والعرض (W) كبيانات حقيقية.**
٢- **المخرجات هي مساحة المستطيل (AREA) كقيمة حقيقية كذلك.**
٣- **أما عمليات المعالجة : فهي قانون حساب مساحة المستطيل المذكور في السؤال:** AREA = L * W
٤- **والمخرجات الناتجة من هذا البرنامج هي المساحة المحسوبة.**

٢.٦.١.٢. "ثانياً : تصميم البرنامج" (Program Design):

يقصد بتصميم البرنامج: تحديد المواصفات والخطوات الدقيقة والمرتبة منطقياً، والتي تم فهمها ودراستها في الخطوة الأولى، ويتم ذلك باستخدام عدة طرق منها:

١- "**الخوارزميات**" (Algorithms).
٢- "**خريطة التدفق**" (Flowchart).

٢.٦.١.٢.١. "الخوارزميات" (Algorithms):

يمكن تعريف الخوارزمية على أنها مجموعة من القواعد والعمليات المعرفة جيداً لحل مشكلة ما في عدد محدد من الخطوات. وقد سميت الخوارزمية بهذا الاسم نسبة إلى عالم الرياضيات المسلم (محمد بن موسى الخوارزمي) – المتوفي سنة ٨٢٥م، وصاحب كتاب الجبر والمقابلة. وهو أول من استعمل طريقة الخوارزميات في المعادلات الجبرية، كما يوضح ذلك المثال التالي:

مثال: اكتب خطوات الخوارزمية لإيجاد قيمة x والتي تحقق المعادلة التالية:

$x + 3 = 6$

الحل : خطوات الخوارزمية هي :

١- **اجعل قيمة x تساوي صفراً، أي:** $x = 0$.
٢- **أحسب قيمة x + 3 : إذا كانت قيمة x + 3 مساوية للقيمة ٦، انتقل للخطوة ٣ ،وإلا قم بما يلي:**
  - **زد قيمة x بمقدار ١، أي:** $x = x + 1$.
  - **عد إلى بداية الخطوة ٢.**
٣- **توقف عن التكرار.**
٤- **اطبع قيمة x.**
٥- **انتهى البرنامج.**

ولكي تكون خطوات الخوارزمية سليمة لا بد أن تحتوي على ثلاث خواص أساسية، كما يوضح ذلك التعريف، وهي:

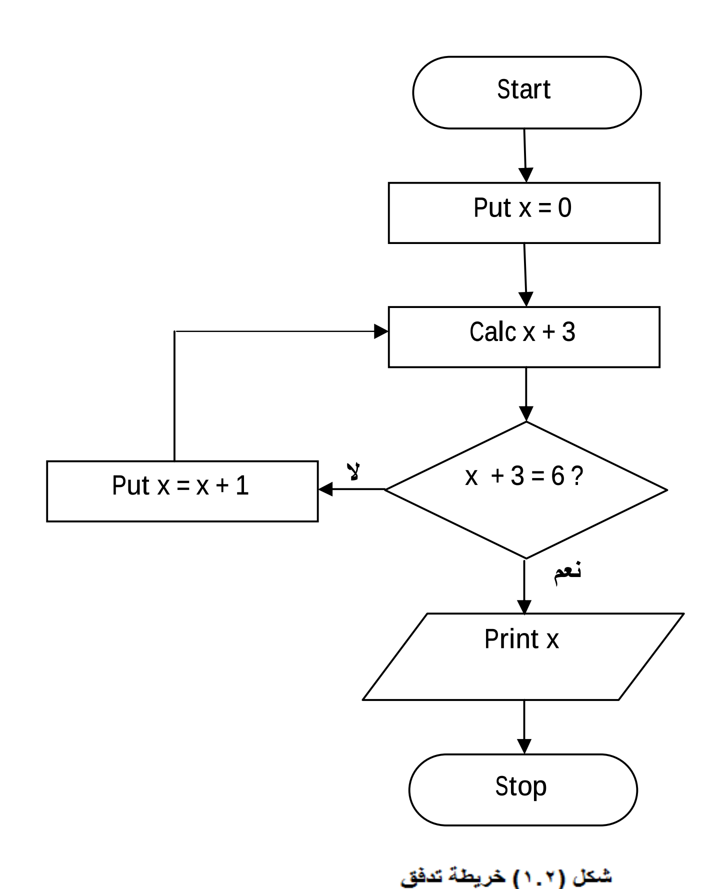
١- كل خطوة يجب أن تكون معرفة جيداً دون أي غموض وبعبارات دقيقة.
٢- أن تتوقف العمليات بعد عدد محدد من الخطوات.
٣- أن تؤدي العمليات بمجملها إلى حل المسألة الحل الصحيح.

وبعد أن نتأكد من أن الخوارزمية تحقق جميع هذه الخواص، وقبل أن نقوم بترجمة هذه الخطوات إلى لغة من لغات البرمجة، يفضل أن نقوم برسم مخطط انسيابي أو خريطة تدفق لهذه الخوارزمية. فما هي المخططات الانسيابية أو خرائط التدفق؟

٢.٢.١.٦.٢. "خريطة التدفق" (Flowchart):

خرائط التدفق أو المخططات الانسيابية هي عبارة عن تمثيل بياني أو رسومي للخوارزمية. ولتمثيل المخططات الانسيابية نحتاج إلى مجموعة من الرموز والأشكال الهندسية، والتي يستخدم كل شكل منها للدلالة على عمل معين، وسنتطرق إليها بالتفصيل في الصفحات القادمة.

والآن سنقوم بتمثيل الخوارزمية في المثال السابق في شكل خريطة تدفق أو مخطط انسيابي، كما يلي:



شكل (١.٢) خريطة تدفق

٢.٦.١.٣. "ثالثاً : كتابة شفرة البرنامج" (Program Coding):

بعد الانتهاء من تصميم البرنامج يتم اختيار إحدى لغات البرمجة المناسبة لـ "كتابة أوامر البرنامج" (Coding)، وذلك بالاستعانة بخريطة التدفق والخوارزمية المكتوبة.

ويجب عن كتابة البرنامج إتباع قواعد صياغة لغة البرمجة المستخدمة، حيث أن لكل لغة برمجة – كما سنتعلم في الفصل التالي– مجموعة من القواعد الخاصة. ولا يعمل البرنامج إذا كان هناك أخطاء إملائية أو ما يطلق عليها "أخطاء صياغية" syntax errors.

٢.٦.١.٤. "رابعاً : اختبار البرنامج وتصحيحه" (Program Testing and Debugging):

يسمى البرنامج بعد كتابته بإحدى لغات البرمجة بـ "البرنامج المصدري" (Source Program). ولا يتم تنفيذ مباشرة على الحاسوب، بل تتم ترجمته أولاً إلى برنامج مكتوب بلغة الآلة، أو ما يمكن أن نطلق عليه بـ "البرنامج الهدفي" (Object Program)، والذي يعتبر ملف قابل للتنفيذ على جهاز الحاسوب. وتسمى عملية التحويل من البرنامج المصدري إلى البرنامج الهدفي بـ "الترجمة" (Compilation)، ويقوم بها برنامج يسمى "المترجم" (Compiler)، وسنتطرق لتفاصيل أكثر حول هذا الموضوع في الفصل القادم.

٢.٦.١.٥. "خامساً : توثيق البرنامج" (Program Documenting):

في هذه المرحلة تتم كتابة وصف تفصيلي لدورة كتابة البرنامج. ويشمل هذا التوثيق أصل المشكلة، ووصف لخطوات الحل وخرائط الحل وتعليمات التشغيل ومتطلبات التشغيل والمدخلات والمخرجات وكيفية التحكم في البرنامج في المواقف المختلفة.

٢.٦.٢. مواصفات برامج الحاسوب الجيدة

من المعلوم أن جهاز الحاسوب عبارة عن آلة صماء، مثله مثل التلفاز أو الفيديو، لا يمكن الاستفادة منه إلا من خلال البرامج التي يتم تشغيلها عليه. لذا فإن جزءاً كبيراً من فاعلية الحاسوب في التعليم يعتمد على جودة البرامج التعليمية المتوفرة. فإذا كانت البرامج جيدة، أمكن الاستفادة منها، أما إذا الأمر غير ذلك فإن هذا يحد من الفائدة المرجوة من استخدام الحاسب في التعليم. لذا فإن برامج الحاسوب لا بد وأن تشتمل على المواصفات الجيدة التي تجعل منه أداة مساعدة في التعليم وفي جميع مجالات الحياة المختلفة. وفيما يلي تعداد لأهم مواصفات برنامج الحاسوب الجيد:

١- خلو المحتوى من الأمور المخلة بالدين أو الأخلاق أو الأعراف أو التقاليد.
٢- أن يكون المحتوى صحيحاً وذا قيمة علمية.
٣- خلوه من الأخطاء الفنية.
٤- أن يكو ن البرنامج سهل الاستخدام.
٥- أن يصدر النتائج المطلوبة بسرعة عالية وكفاءة كبيرة.
٦- أن يوظف البرنامج قدرات الحاسوب المختلفة (الألوان ، الأصوات ، الحركة ، ..).
٧- أن يشمل جوانب شد الانتباه والتحدي وشحذ الهمم.

٧.٢. الخوارزميات

"الخوارزمية" (Algorithm) هي عبارة عن مجموعة من الخطوات التي تولد سلسلة منتهية من العمليات الحسابية الأساسية تقود إلى حل مشكلة معينة.

وقد تحدثنا عن الخوارزميات وأنها إحدى مراحل تطوير البرامج الحاسوبية. إذاً، فالخوارزمية تتكون من خطوات مرتبة، بعضها إثر بعض، وكل خطوة تعتبر بنفسها وحدة من وحدات البناء الكامل للخوارزمية . ويختلف حجم هذه الخطوات باختلاف الخوارزميات، واختلاف الأشخاص الذين يقومون بتنفيذ تلك الخطوات. مما يعني أن المشكلة الواحدة قد تحل بأكثر من خوارزمية وكلها يؤدي إلى نفس النتيجة رغم اختلاف كفاءة هذه الخوارزميات من حيث السرعة والدقة والمساحة التخزينية. وهناك قانون يصف هذه النقطة، وهو "قانون النهاية الواحدة" (Law of Equifinality).

١.٧.٢. قانون النهاية الواحدة Equi-finality Law

وهو قانونٌ ينص على أنه يمكن تحقيق الهدف الواحد من خلال عدة خطوات أو عدة مسارات مختلفة. وكلما كان استيعاب المبرمج للمشكلة أوضح وأدق، كان التصور الذي وضعه أسرع وأكفأ. والمعلوم بداهةً أن خبرة المبرمج تزداد من خلال الممارسة المستمرة والتدريب المتواصل، حتى يتمكن المبرمج من هذه المهنة الشيقة. وسوف نتعرف في هذا الفصل على كيفية كتابة المشكلة بعد تحليلها في شكل خوارزمية مرتبة الخطوات.

٨.٢. المخططات الانسيابية Flowcharts

وتسمى أيضاً "خرائط التدفق" (Flowcharts). وهذه أيضاً قد تحدثنا عنها وقلنا إنها عبارة عن تمثيل رسومي لخطوات الخوارزمية. والمعلوم إن التمثيل في شكل رسومي أو بياني يختصر المعلومة المكتوبة نصياً ويسهل فهمها. ولذا، فإن الكثير من الكتاب يستخدمون الأشكال والرسوم للتوضيح والاختصار عند كتابتهم لمواضيع خصوصاً العلمية منها. إذا فالمخططات الانسيابية جاءت لتوضيح واختصار خطوات الخوارزمية. فبمجرد النظر إلى المخطط الانسيابي، يمكنك فهم ماهية المشكلة وطبيعة حلها.

١.٨.٢. فوائد المخططات الانسيابية Flowcharts Benifits

وتكمن فوائدها في النقاط التالية:

١- تعطي صورة متكاملة للخطوات المطلوبة لحل المشاكل في ذهن المبرمج، بحيث تمكنه من الإحاطة الكاملة بكل أجزاء المشكلة من بدايتها وحتى نهايتها.

٢- توضيح الطريقة التي يمر بها البرنامج من المدخلات أو البيانات ومن ثم المعالجة، وأخيراً مخرجات ونتائج البرنامج.

٣- توثيق منطق البرنامج للرجوع إليه عند الحاجة، وذلك بغرض إجراء أي تعديلات على البرنامج أو اكتشاف الأخطاء التي تقع عادةً في البرامج، وخصوصاً الأخطاء المنطقية.

٤- تيسر للمبرمج أمر إدخال أي تعديلات، في أي جزء من أجزاء المشكلة، بسرعة، وبدون الحاجة لإعادة دراسة المشكلة برمتها من جديد.

٥- في المشاكل التي تكثر فيها الاحتمالات والتفرعات، يصبح أمر متابعة دقائق التسلسل أمراً شاقاً على المبرمج، إذا لم يستعن بمخطط تظهر فيه خطوات الحل الرئيسية بشكل واضح.

٦- تعتبر رسوم خرائط التدفق المستعملة في تصميم حلول بعض المشاكل، مرجعاً في حل مشاكل أخرى مشابهة، ومفتاحاً لحل مشاكل أخرى جديدة لها علاقة مع المشاكل القديمة المحلولة. فيمكن تشبيه رسوم خرائط التدفق بالرسوم التخطيطية التي يقوم بها المهندس المعماري، حيث أنه يستفيد من المخططات القديمة في بناء مخططات جديدة عن طريق التحديث بالإضافة أو الحذف أو التعديل على المخططات القديمة. وهذا يعتبر عاملاً من عوامل السرعة في تطوير وبناء النظم البرمجية.

٢.٨.٢. أنواع المخططات الانسيابية

بشكلٍ عام، يمكن القول بأن هناك نوعين رئيسين من خرائط التدفق، وهما:

١- "خرائط تدفق النظام" (System Flowcharts).

٢- "خرائط تدفق البرنامج" (Program Flowcharts).

١.٢.٨.٢. "أولاً : خرائط تدفق النظام" (System Flowcharts):

يستخدم هذا النوع من الخرائط عند تصميم الأجهزة الهندسية، في المصانع وغيرها، والتي تستعمل أنظمة تحكم ذاتية، مثل العوامة في خزانات المياه، وإشارات المرور الضوئية، وأجهزة ضبط الضغط ودرجات الحرارة في أبراج تقطير البترول. فتعتبر خرائط التدفق هنا بمثابة المخطط الكامل الذي يبين ترتيب وعلاقة ووظيفة كل مرحلة بما قبلها وبما بعدها، داخل إطار النظام المتكامل. ويمكن تلخيص الدور الذي تقدمه هذه الخرائط بما يأتي:

١- تبين موقع كل خطوة من الخطوات الأخرى المكونة للنظام، بحيث يسهل اكتشاف أي خلل يحدث في النظام كله بمجرد النظر، مما ييسر عمليات صيانة الأجهزة، وبأقل التكاليف.

٢- تسهل إجراء التعديلات التي قد تطرأ مستقبلاً على برنامج النظام في أي موقع منه.

٣- بيان تفاصيل البيانات المطلوب إدخالها إلى النظام.

٤- بيان تفاصيل أنواع النتائج المتوقعة أو المطلوبة من البرنامج المعد للنظام.

٥- بيان طرق ربط النظام، ببقية الأنظمة الموجودة في المؤسسة.

٢.٢.٨.٢. "خرائط تدفق البرنامج" (Program Flowcharts):

ويستعمل هذا النوع من الخرائط، لبيان الخطوات الرئيسية التي توضع لحل مشكلة ما. وذلك بشكل رسوم اصطلاحية تبين العلاقات المنطقية بين سائر خطوات الحل. وموقع وإطار كل منها في إطار الحل الشامل للمسألة. وهذا النوع من الخرائط سيكون محور دراستنا في القادم من الصفحات.

٣.٨.٢. الرموز الاصطلاحية للمخططات الانسيابية

فيما يلي أهم الرموز الاصطلاحية المستخدمة لرسم خرائط تدفق البرنامج:

الرمز	المصطلح	استخدامه
	Terminal	يستخدم في بداية ونهاية المخطط الانسيابي
	Input / Output	يستخدم لتبيين أي عمليات إدخال أو إخراج
	Processing	يستخدم لتبيين أي عمليات معالجة تتم من قبل الحاسوب
	Comment	يستخدم لكتابة أي تعليقات توضيحية زائدة لشرح شيء ما في المشكلة
	Flow line	خط تدفق، يبين جهة انسياب البيانات بين الرموز المختلفة للمخطط
	Decision	يستخدم لتوضيح أي نقطة في البرنامج يتم فيها اتخاذ قرار سواءً بالتفرع أو بالتكرار
	On-page connector	رابط يستخدم لربط أجزاء المخطط في نفس الصفحة
	Off-page connector	رابط يستخدم لربط أجزاء المخطط في صفحات مختلفة

٩.٢. تطبيقات على الخوارزميات والمخططات الانسيابية Algorithms and Flowcharts Execises

سنقوم في هذا الجزء من الفصل، بتقديم بعض الأمثلة التوضيحية التي من خلالها نشرح كيفية تحليل المشكلة، ومن ثم كيفية كتابتها في شكل خوارزمية ، وكذا كيفية تمثيل هذه الخوارزمية بشكل رسومي عن طريق المخطط الانسيابي. وفي الفصل التالي – عند دراستنا لمبادئ البرمجة بلغة C++ – سنتعلم أسس وقواعد هذه اللغة والتي من خلالها سنتمكن من تحويل هذه التطبيقات إلى برامج عملية بلغة C++ قابلة للتنفيذ على جهاز الحاسوب بغرض الحصول على المخرجات الفعلية. وما سنقوم به في هذا الفصل والفصل الذي يليه، إنما هو تطبيق لمراحل تطوير البرامج الحاسوبية التي تناولنها آنفاً.

ولكن قبل ذلك، سنقوم بعملية تصنيف لهذه التطبيقات قبل البدء بدراستها. حيث أننا يمكن أن نصنف هذه التطبيقات إلى ثلاثة أصناف رئيسية:

١- "مشاكل تسلسلية" (Sequential Problems).
٢- "مشاكل الاختيار أو التفرع" (Selection Problems).
٣- "مشاكل التكرار أو الدوران" (Looping Problems).

١.٩.٢. "**مشاكل تسلسلية**" (Sequential Problems)

وفيها يتم ترتيب خطوات الحل بشكل سلسلة مستقيمة من بداية البرنامج وحتى نهايته. حيث أنها تخلو من أي نقاط تفرع تؤدي إلى تعدد مسارات الحل، ومن أي دورانات تؤدي إلى تكرار مجموعة من الخطوات. وفي المثال التالي توضيح لهذا المفهوم:

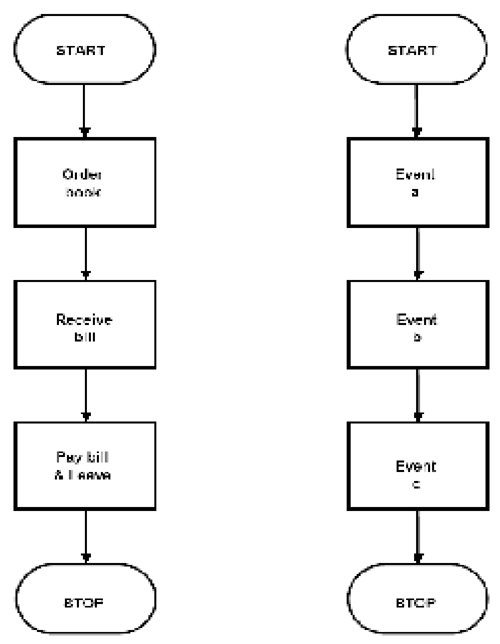
مثال: اكتب خوارزمية وخريطة تدفق لبرنامج يمثل عملية شراء كتاب من مركز بيع.

الحل: الخوارزمية يمكن كتابتها باختصار كما يلي:

١- اطلب الكتاب.
٢- استلم الفاتورة.
٣- ادفع القيمة وانصرف.

ويمكن رسم خريطة التدفق كما يلي:

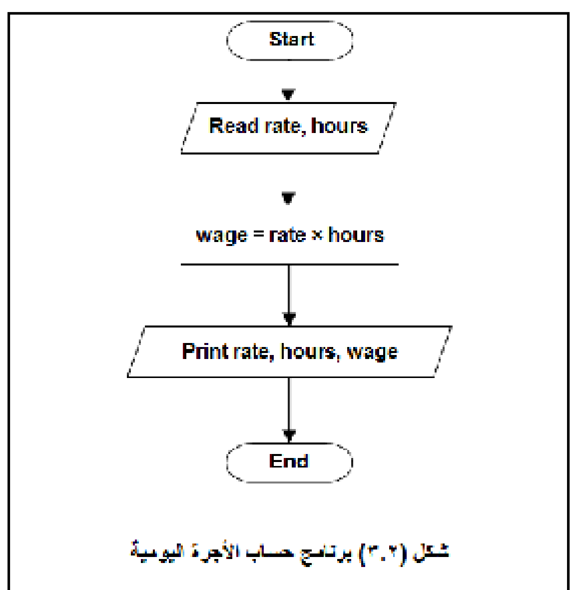
نلاحظ من خلال الشكل أنه في المشاكل التسلسلية يوجد مجموعة أحداث (a و b و c و ...). يتم تنفيذ هذه الأحداث بشكل تتابعي أو تسلسلي حتى نهاية البرنامج.



شكل (٢.٢) سيناريو البرنامج التسلسلي

مثـــــــــــال ٢.١.١.٩

ارسم خريطة سير البرنامج (flow chart) لإيجاد الأجرة اليومية wage لعامل اعتماداً على عدد الساعات hours التي يعملها في اليوم، وأجرة الساعة الواحدة rate.



شكل (٣.٢) برنامج حساب الأجرة اليومية

الحـــــــل:

الشكل (٣.٢) يبين خريطة سير البرنامج (flow chart) لحل هذه المسألة كما يلي:

١- اقرأ قيمة الأجرة rate، وعدد الساعات التي عملها الموظف hours.
٢- أحسب الأجرة اليومية من المعادلة التالية: $wage = rate \times hours$.
٣- أطبع كلاً من hours، rate، و wage.
٤- انتهى البرنامج.

مثـــــــــــال ٢.١.٢.٩

ارسم خريطة سير البرنامج (flow chart) لإيجاد مساحة ومحيط دائرة نصف قطرها R.

الحـــــــل:

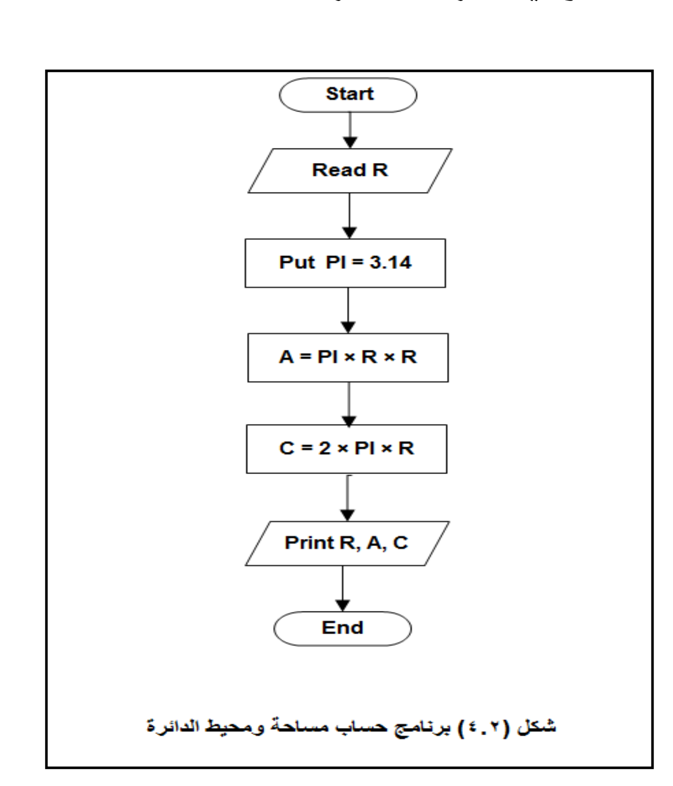
مساحة الدائرة = πr^2

محيط الدائرة = $2\pi r^2$

حيث π = النسبة التقريبية وقيمتها العددية ثابتة وتساوي بالتقريب 3.14، بينما R متغير يمثل قيمة نصف القطر.

وحل هذه المسألة كما يلي:

١- اقرأ قيمة R.
٢- ضع قيمة $\pi = PI = 3.14$
٣- أحسب مساحة الدائرة A من المعادلة $A = \pi r^2$
٤- أحسب محيط الدائرة C من المعادلة $C = 2\pi r$
٥- أطبع قيم كل من: R, A, C.
٦- انتهى البرنامج.



شكل (٤.٢) برنامج حساب مساحة ومحيط الدائرة

خريطة سير البرنامج التي توضح حل هذه المسألة موضحة في الشكل (٤.٢).

مثـــــــــال ٣.١.٩.٢

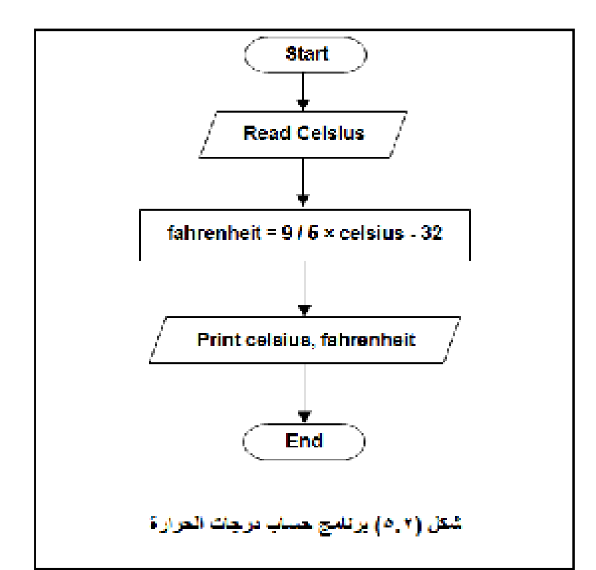
اكتب برنامج يقوم باستقبال درجة الحرارة بالنظام المئوي (Celsius) ويحولها إلى النظام الفهرنهايتي (Fahrenheit) ، علماً بأن معادلة التحويل هي:

$.fahrenheit = \frac{9}{5} \times celsius - 32$



الحـــــل:

الشكل (٥.٢) يبين خريطة سير البرنامج (flow chart) لحل هذه المسألة كما يلي:



شكل (٥.٢) برنامج حساب درجات الحرارة

١- اقرأ درجة الحرارة بالنظام المئوي (Celsius).
٢- أحسب درجة الحرارة بالنظام الفهرنهايتي (Fahrenheit) من العلاقة:

$.fahrenheit = \frac{9}{5} \times celsius - 32$

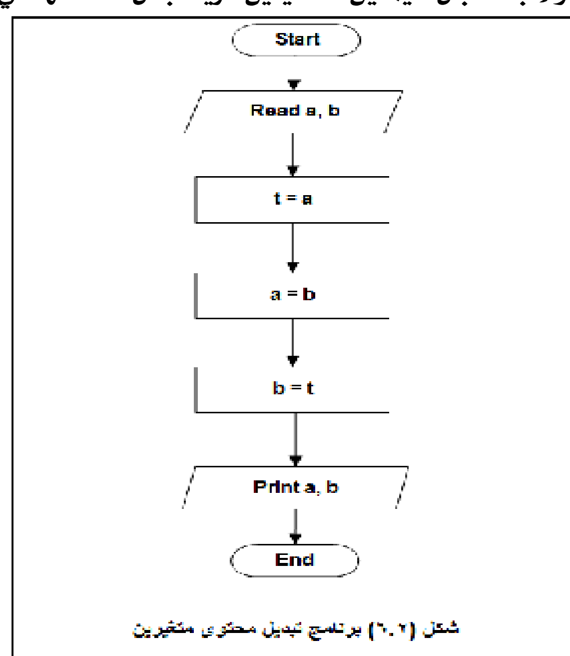
٣- اطبع قيم Celsius و Fahrenheit.
٤- انتهى البرنامج.

مثـــــــــال ٤.١.٩.٢

ارسم خريطة سير البرنامج (flow chart) لبرنامج يقوم باستقبال قيمتين عدديتين، ويستبدل أماكنها في الذاكرة .

الحـــــل:

الشكل (٦.٢) يبين خريطة سير البرنامج (flow chart) لحل هذه المسألة كما يلي:



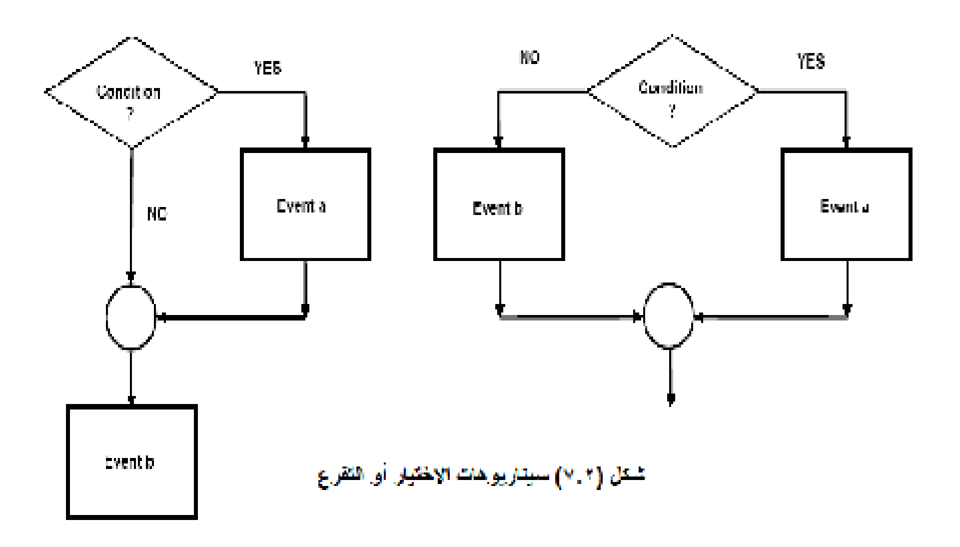
شكل (٦.٢) برنامج تبديل محتوى متغيرين

١- اقرأ قيمة العددين، وليكونا: a، b.
٢- احفظ قيمة a في متغير مؤقت، وليكن t: $.t = a$
٣- احفظ قيمة b في المتغير a: $.a = b$
٤- احفظ قيم المتغير المؤقت t في b: $.b = t$
٥- اطبع قيم a و b.
٦- انتهى البرنامج.

٢.٩.٢. "**مشاكل الاختيار أو التفرع**" (Selection Problems)

قد نحتاج لاتخاذ قرار ما حيال مشكلة معينة، مما قد يؤدي إلى تفرع أو تعدد مسارات الحل. بحيث يكون عندنا أكثر من مسار للحل تعتمد على تحقق شرط معين. ففي كل مرة يتم فيها تنفيذ البرنامج يتم اختبار هذا الشرط، وعلى ضوء هذا الاختبار يتم تحديد المسار الذي ينبغي سلوكه وتجاهل بقية المسارات.

وبشكل عام، فإن مشاكل الاختيار أو التفرع، يمكن أن تأخذ أحد الشكلين التاليين:



شكل (٢.٧) سيناريوهات الاختيار أو التفرع

ففي الشكل الأول من جهة الشمال يمكن ملاحظة أنه إذا كان جواب الشرط "نعم"، فإن الحدث الذي سيتم تنفيذه هو a **ثم يليه** b **(أي أن كلا الحدثين سينفذان). أما إذا كان جواب الشرط "لا"، فإن الحدث الذي سيتم تنفيذه هو** b **فقط (أي أن** a **سيتم تجاهله).**

أما في الشكل الآخر فيمكن ملاحظة أنه إذا كان جواب الشرط "نعم"، فإن الحدث التالي في التنفيذ سيكون a**، (وسيتم تجاهل الحدث** b**). أما إذا كان جواب الشرط "لا"، فإن الحدث التالي في التنفيذ سيكون** b**، (وسيتم تجاهل الحدث** a**).**

يسمى النوع الأول من جمل الاختيار بـ (single-selection statement**) أو "جملة الاختيار الأحادية"، بينما يسمى النوع الثاني من جمل الاختيار بـ (**double-selection statement**) أو "جملة الاختيار الثنائية". ويجدر بنا التنبيه إلى أن هناك نوع ثالث من جمل الاختيار هو (**multiple-selection statement**) أو "جملة الاختيار المتعددة" وسنأتي عليها عند دراستنا لجملة** switch **في لغة** C++ **في الفصل القادم إن شاء الله.**

مثـــــــــــــال ١.٢.٩.٢

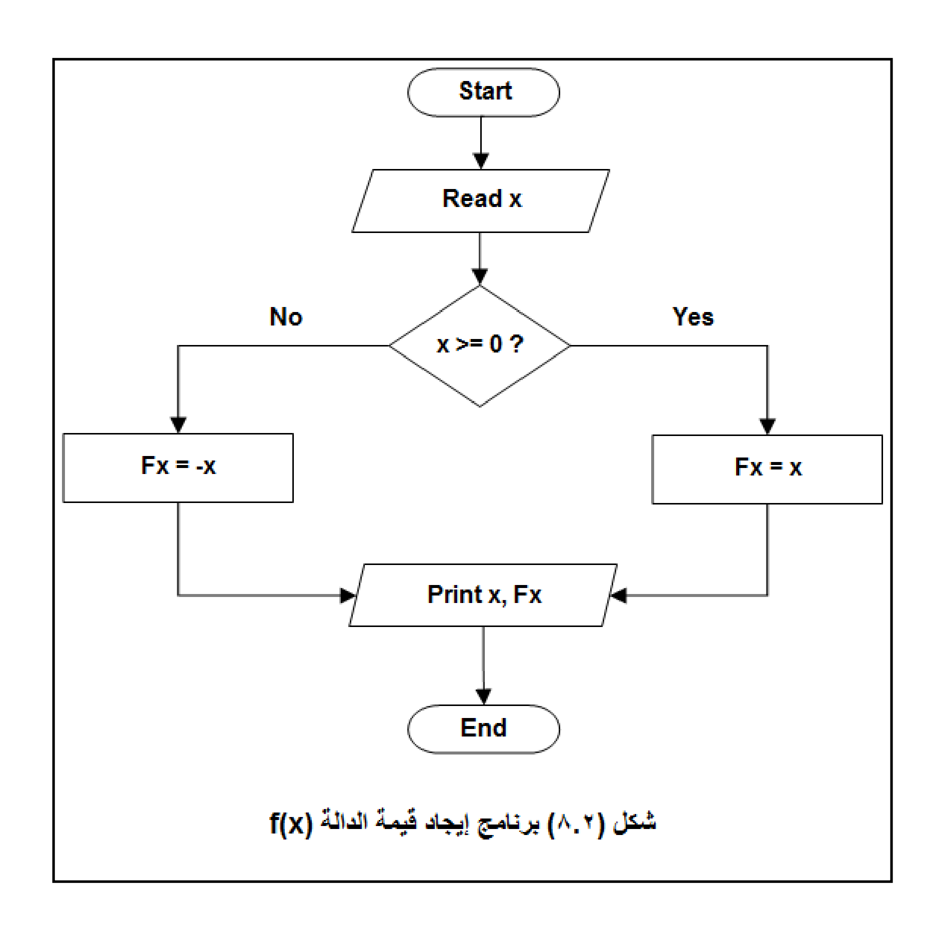
ارسم خريطة سير البرنامج (flow chart) لإيجاد قيمة الدالة F(x) المعرفة كما يلي:

$$f(x) = \begin{cases} x & x \geq 0 \\ -x & x < 0 \end{cases}$$

الحـــــــــــل:

الشكل (٨.٢) يبين خريطة سير البرنامج (flow chart) لحل هذه المسألة كما يلي:

١- اقرأ قيمة المتغير x.
٢- إذا كانت x أكبر من أو تساوي صفراً، اذهب إلى الخطوة ٣، وإلا فاذهب إلى الخطوة ٤.
٣- أحسب قيمة f(x) من الدالة $f(x) = x$ ثم اذهب إلى الخطوة ٥.
٤- أحسب قيمة f(x) من الدالة $f(x) = -x$.
٥- اطبع قيمة كل من x و f(x).
٦- انتهى البرنامج.



شكل (٨.٢) برنامج إيجاد قيمة الدالة f(x)

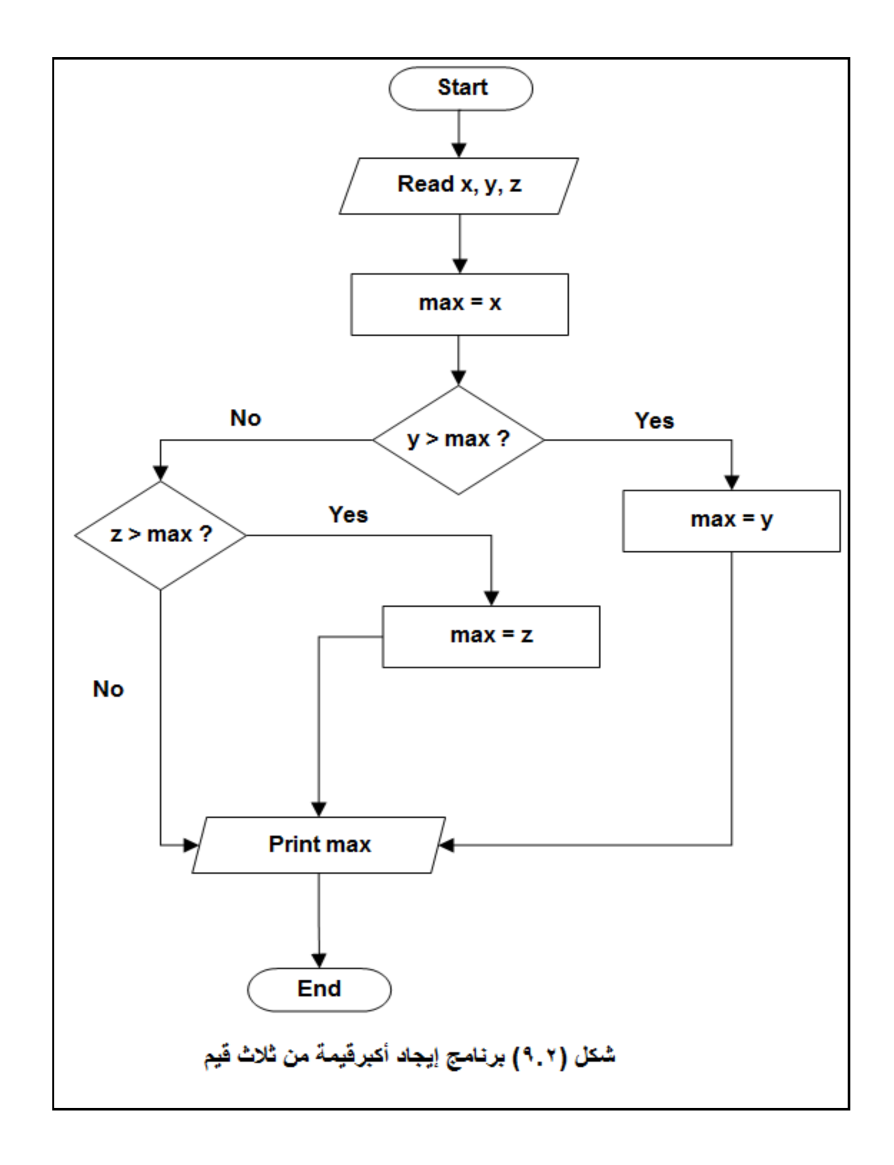
مثـــــــــــــــــال ٢.٢.٩.٢

ارسم خريطة سير البرنامج (flow chart) لبرنامج يستقبل ثلاث قيم ويطبع أكبرها .

الحـــــــــــل:

خطوات الحل مبينة في الشكل (٩.٢) ، وهي:

١- اقرأ قيم هذه الأعداد ، ولتكن: x، y، z.
٢- افرض أن أول عدد هو أكبر عدد max، أي: $max = x$.
٣- إذا كانت قيمة y أكبر من max، أجعل $max = y$، والا إذا كانت قيمة z أكبر من max، اجعل $max = z$.
٤- أطبع قيمة max.
٥- انتهى البرنامج.



شكل (٩.٢) برنامج إيجاد أكبرقيمة من ثلاث قيم

مثـــــــــــــــال ٣.٢.٩.٢

ارسم خريطة سير البرنامج (flow chart) لبرنامج يقوم بحساب تكلفة فاتورة تلفون حسب القواعد الثلاث التالية:

- **أول ٥٠ دقيقة: كل دقيقة بـ ١٠ ريالات.**
- **الـ ١٠٠ دقيقة التالية: كل دقيقة بـ ١٥ ريال.**
- **ما زاد على ذلك: كل دقيقة بـ ٢٠ ريال.**

<u>الحـــــــــــل:</u>

خطوات الحل مبينة في الشكل (١٠.٢)، وهي:

١- **اقرأ عدد الدقائق المستهلكة، وليكن** minutes.

٢- **إذا كانت** minutes **أقل من أو يساوي ٥٠، اذهب إلى الخطوة ٣، والا إذا كانت** minutes **أقل من أو يساوي ١٥٠، اذهب إلى الخطوة ٤، والا اذهب إلى الخطوة ٥.**

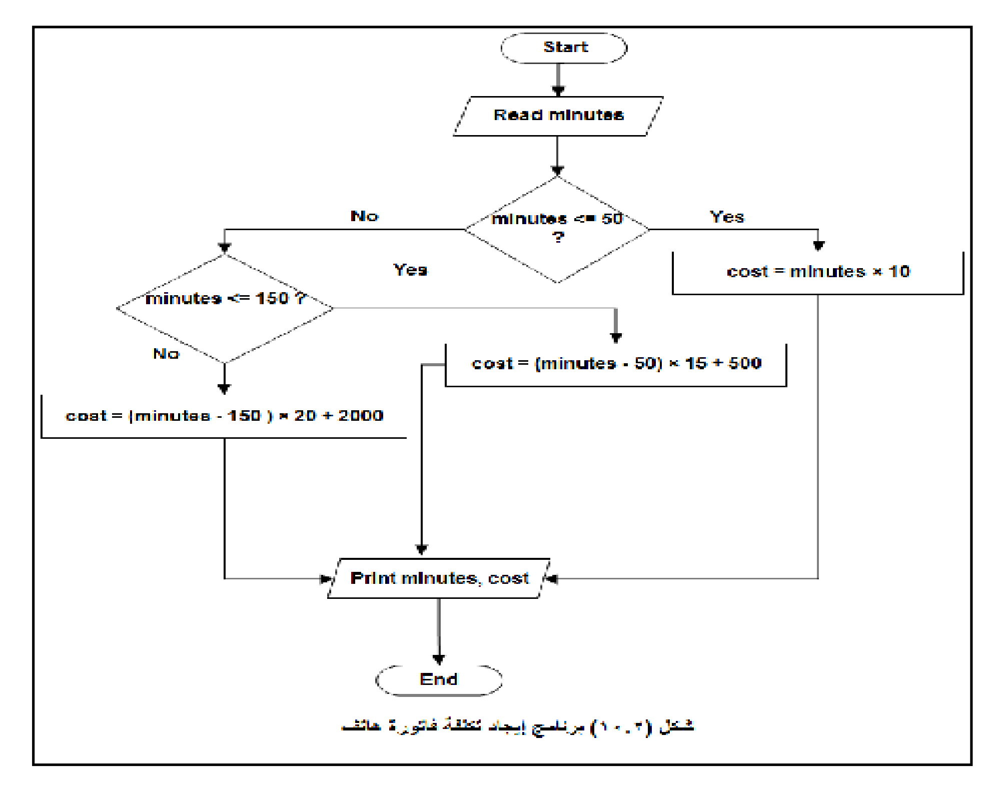
٣- **أحسب تكلفة الفاتورة** cost**، من المعادلة:** $cost = minutes \times 10$**، ثم اذهب إلى الخطوة ٦.**

٤- **أحسب تكلفة الفاتورة** cost**، من المعادلة:** $cost = (minutes - 50) \times 15 + 500$**، ثم اذهب إلى الخطوة ٦.**

٥- **أحسب تكلفة الفاتورة** cost**، من المعادلة:** $cost = (minutes - 150) \times 20 + 2000$.

٦- **إطبع قيمة كلاً من** minutes **و** cost.

٧- **انتهى البرنامج.**



شكل (١٠.٢) برنامج إيجاد تكلفة فاتورة هاتف

مثـــــــــال ٤.٢.٩.٢

ارسم خريطة سير البرنامج (flow chart) لإيجاد جذور المعادلة من الدرجة الثانية في مجهول واحد، علماً بأن الشكل العام لهذا النوع من المعادلات هو: $ax^2 + bx + c = 0$

الحـــــــــل:

يمكن حل هذه المعادلة باستخدام القانون العام اعتماداً علي قيمة دلتا Δ والتي تحسب من العلاقة التالية: $\Delta = b^2 - 4ac$، وللحل ثلاث حالات هي:

- **إذا كانت قيمة Δ أكبر من الصفر، فللمعادلة جذران مختلفان هما:**

$$x1 = \frac{-b + \sqrt{\Delta}}{2a}$$
$$x2 = \frac{-b - \sqrt{\Delta}}{2a}$$

- **أما إذا كانت قيمة Δ تساوي الصفر، فللمعادلة جذران متساويان هما:**

$$x1 = x2 = \frac{-b}{2a}$$

- **أما فيما عدا ذلك (أي عندما تكون قيمة Δ أقل من الصفر)، فليس للمعادلة حل في مجموعة الأعداد الحقيقية.**

وبالتالي ستكون خطوات الحل كما هي مبينة في الشكل (١١.٢)، وهي:

١- **اقرأ قيم المعاملات a، b، و c.**
٢- **أحسب قيمة Δ من المعادلة: $delta = b^2 - 4ac$.**
٣- **إذا كانت قيمة Δ أكبر من الصفر، اذهب إلى ٤، والا إذا كانت قيمة Δ تساوي الصفر، اذهب إلى ٥، والا اذهب إلى ٦.**
٤- **احسب جذري المعادلة كما يلي:**

$$x1 = \frac{-b + \sqrt{delta}}{2a} \quad , \quad x2 = \frac{-b - \sqrt{delta}}{2a}$$

، ثم اذهب إلى الخطوة ٧.

٥- **احسب جذري المعادلة كما يلي:**

$$x1 = x2 = \frac{-b}{2a}$$

، ثم اذهب إلى الخطوة ٧.

٦- **اطبع عبارة *'No solutions'*، ثم اذهب إلى الخطوة ٨.**
٧- **اطبع قيم كلاً من *x1* و *x2*.**
٨- **انتهى البرنامج.**

Start

Read a, b, c

delta = b × b - 4ac

delta > 0 ?

No

Yes

x1 = (-b + sqrt(delta)) / 2a

x1 = (-b - sqrt(delta)) / 2a

delta = 0 ?

Yes

No

x1 = x2 = -b / 2a

Print 'No solutions'

Print x1, x2

End

شكل (١١.٢) برنامج إيجاد جذور معادلة الدرجة الثانية في مجهول

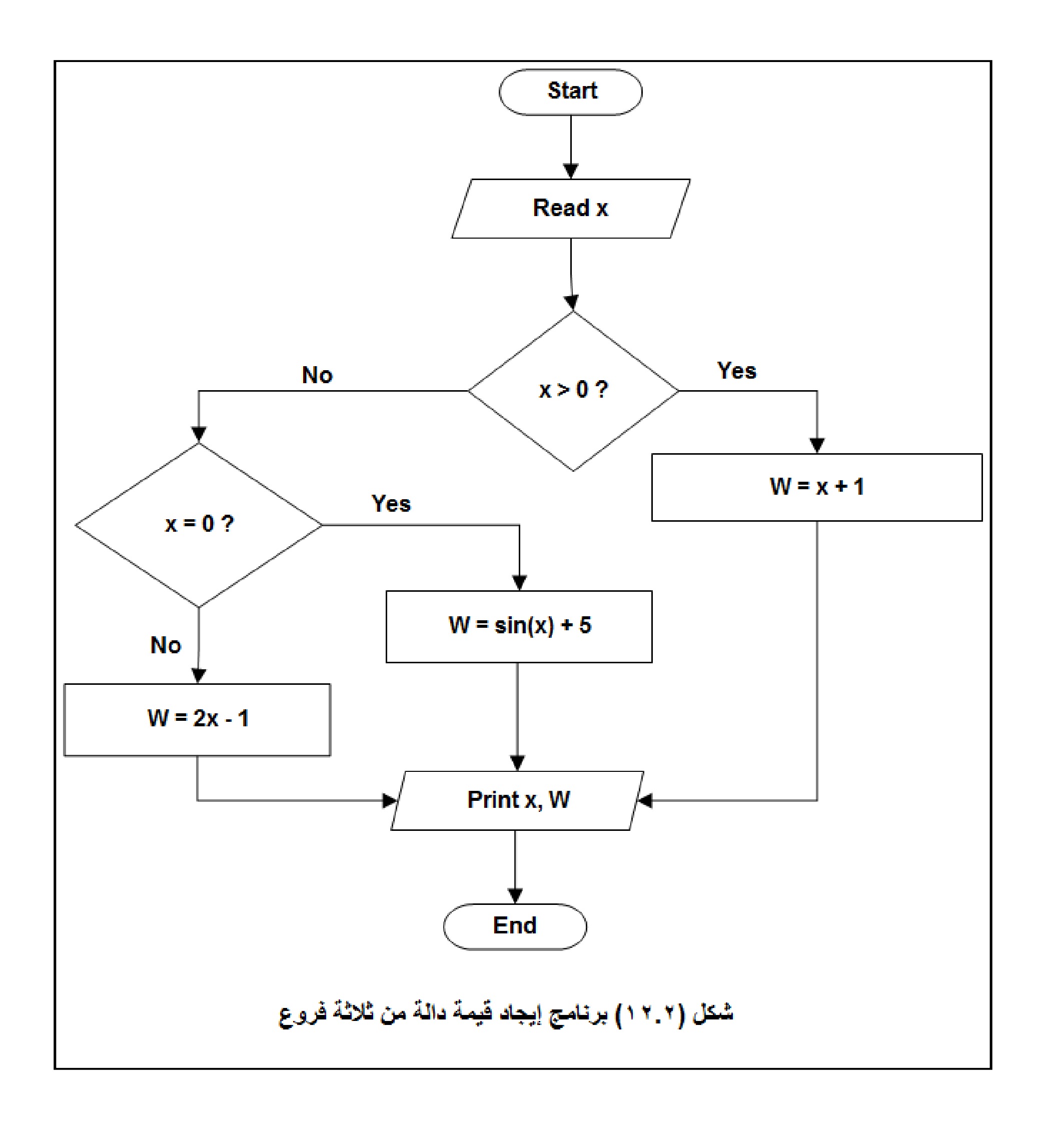


ارسم خريطة سير البرنامج (flow chart) لإيجاد قيمة الدالة W من المعادلات التالية، علماً بأن قيمة المتغير x معلومة:

الحـــــــــــل:

خطوات الحل مبينة في الشكل (١٢.٢)، وهي:

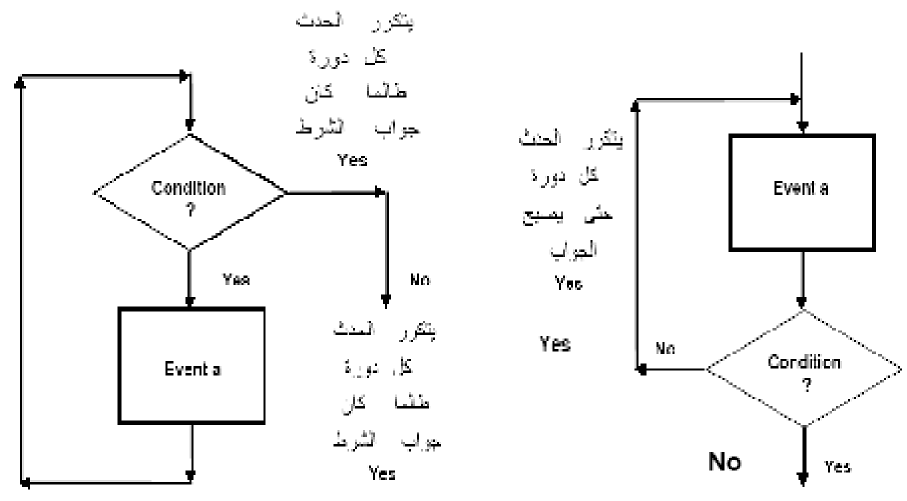
١- اقرأ قيمة x.
٢- إذا كانت x أكبر من الصفر، اذهب إلى الخطوة ٢، وإلا إذا كانت x تساوي الصفر، اذهب إلى الخطوة ٣، والا اذهب إلى الخطوة ٤.
٣- احسب W من المعادلة: $W = x + 1$، ثم اذهب إلى الخطوة ٥.
٤- احسب W من المعادلة: $W = \sin(x) + 5$، ثم اذهب إلى الخطوة ٥.
٥- احسب W من المعادلة: $W = 2x - 1$.
٦- اطبع قيم كلاً من x، W.
٧- انتهى البرنامج.



شكل (١٢.٢) برنامج إيجاد قيمة دالة من ثلاثة فروع

٣.٩.٢. "**مشاكل التكرار أو الدوران**" (Looping Problems)

وهذا النوع يظهر عند الحاجة إلى تكرار حدث أو مجموعة من الأحداث لعدد محدد أو غير محدد من المرات يستمر في الحالتين بتحقق شرط معين يسمى "شرط الاستمرار"، وينتهي التكرار بانتفاء تحقق هذا الشرط. وهناك في العموم نوعان من بنى التكرار.



شكل (١٣.٢) سيناريوهات بنى التكرار (الدوران)

النوع الأول من جهة الشمال يمثل تكراراً يتم فيه اختبار شرط الاستمرار أولاً، ويتكرر تنفيذ الحدث a طالما أن جواب الشرط صائب، ومن هنا يتضح أن هناك احتمال أن لا ينفذ الحدث a مطلقاً، وذلك في حالة أن الشرط لم يتحقق من البداية أصلاً.

النوع الآخر يمثل تكراراً يتم فيه تنفيذ الحدث a أولاً، ثم اختبار شرط الاستمرار ثانياً، ويتكرر تنفيذ الحدث a طالما أن جواب الشرط صائب، ومن هنا يتضح أن أقل احتمال لتنفيذ الحدث a هو مرة واحدة على الأقل،وهي أول مرة، لأنها تمت قبل الدخول على الشرط.

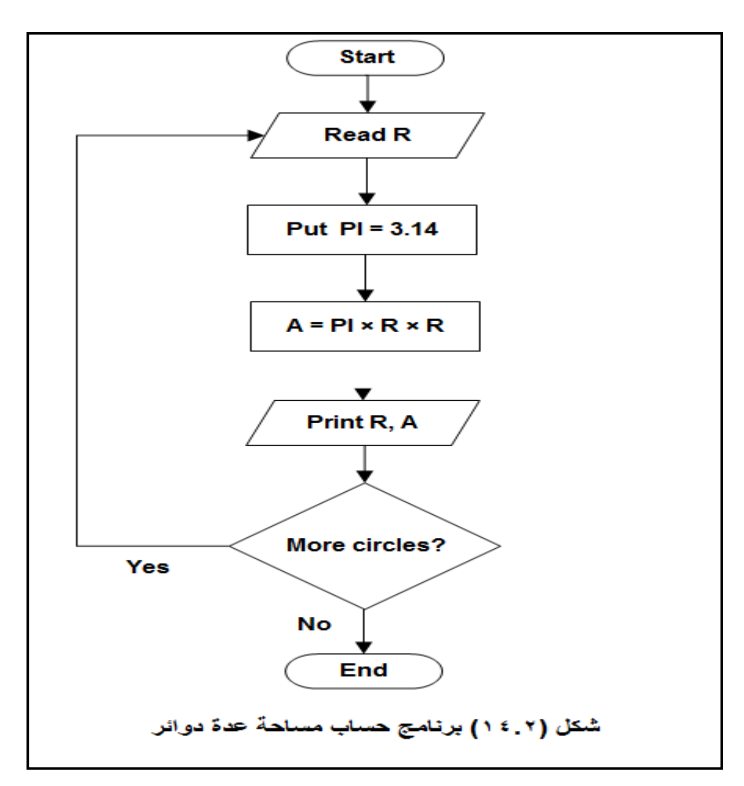
مثـــــــــــــال ١.٣.٩.٢

ارسم خريطة سير البرنامج (flow chart) لإيجاد مساحة مجموعة من الدوائر أنصاف أقطارها معلومة.

الحـــــــــل:

خطوات الحل مبينة في الشكل (١٤.٢)، وهي:

١- اقرأ قيمة R.
٢- ضع قيمة $\pi = PI = 3.14$.
٣- أحسب مساحة الدائرة A من المعادلة $A = \pi r^2$.
٤- أطبع قيم كل من: R, A.
٥- هل هناك المزيد من الدوائر؟
- إذا كان نعم، عد للخطوة ١،
- أما إذا كان لا، فتوقف عن التكرار.
٦- انتهى البرنامج.



شكل (١٤.٢) برنامج حساب مساحة عدة دوائر

مثـــــــــــــال ٢.٣.٩.٢

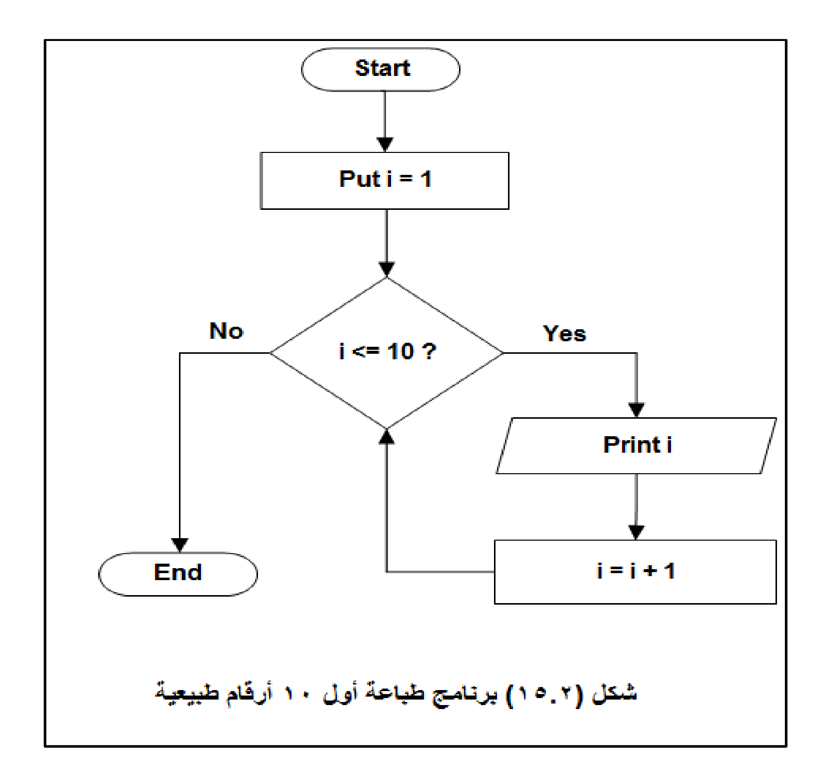
ارسم خريطة سير البرنامج (flow chart) لبرنامج يطبع أول ١٠ أعداد طبيعية.

الحـــــــــل:

في هذا البرنامج نقوم بتعريف عداد يبدأ بـ ١ وينتهي بـ ١٠ وليكن i، و نقوم في كل دورة من دورات الحلقة بطباعة قيمة العداد نفسها ثم زيادتها بمقدار واحد صحيح.

وخطوات الحل مبينة في الشكل (١٥.٢)، وهي كما يلي:

١- اجعل قيمة العداد i تساوي واحد، أي: $i = 1$.
٢- إذا كانت قيمة i لا تزال أقل من أو تساوي ١٠، قم بما يلي، وإلا انتقل إلى الخطوة ٣.
- اطبع قيمة i.
- زد قيمة i بمقدار ١، أي: $i = i + 1$.
- عد إلى بداية الخطوة ٢.
٣- انتهى التكرار.
٤- انتهى البرنامج.



شكل (١٥.٢) برنامج طباعة أول ١٠ أرقام طبيعية

مثـــــــــــــال ٢.٩.٣.٣

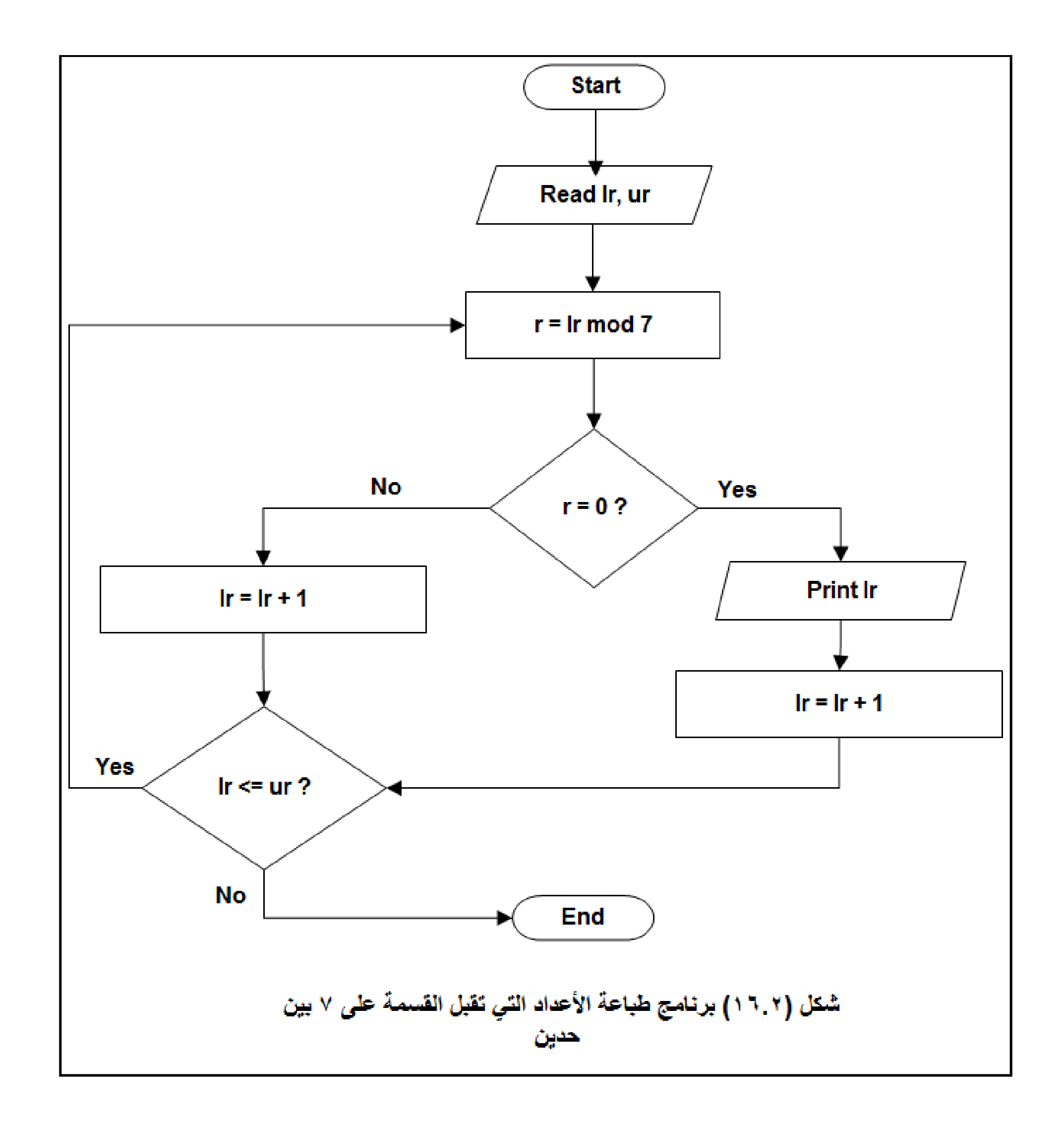
ارسم خريطة سير البرنامج (flow chart) لبرنامج يعمل على الحصول على كل الأعداد التي تقبل القسمة على ٧ ضمن مدى معين ويطبعها.

الحـــــــــــل:

في هذا البرنامج نقوم بإدخال النهاية الصغرى للأعداد، ولتكن lr والنهاية العظمى، ولتكن ur. ثم نقوم بقسمة النهاية الصغرى على ٧ ، إذا كان باقي القسمة صفر نطبع النهاية الصغرى، وإلا نزيد النهاية الصغرى بمقدار ١، ثم نختبرها ، فإذا تجاوزت النهاية العظمى نتوقف، وإلا نعود إلى خطوة القسمة من جديد.

والشكل (١٦.٢) التالي يوضح خطوات الحل، وهي:

١- اقرأ النهاية الصغرى lr، والنهاية العظمى ur.
٢- أحسب باقي قسمة النهاية الصغرى على ٧ واحفظه في متغير r، كما في المعادلة: $r = lr \bmod 7$.
٣- إذا كانت قيمة باقي القسمة r تساوي صفراً، انتقل إلى الخطوة ٤، وإلا انتقل إلى الخطوة ٥:
٤- - اطبع قيمة lr.
- زد قيمة lr بمقدار ١، أي: $lr = lr + 1$. انتقل إلى الخطوة ٦.
٥- زد قيمة lr بمقدار ١، أي: $lr = lr + 1$.
٦- إذا كانت قيمة lr ما تزال أقل من أو تساوي قيمة ur، ارجع للخطوة ٢. وإلا انتقل إلى الخطوة التالية.
٧- توقف عن التكرار.
٨- انتهى البرنامج.



شكل (١٦.٢) برنامج طباعة الأعداد التي تقبل القسمة على ٧ بين حدين

مثـــــــــــــــال ٤.٣.٩.٢

ارسم خريطة سير البرنامج (flow chart) لبرنامج يعمل على الحصول على مجموع كل الأعداد التي تقبل القسمة على ٧ ضمن مدى معين ويطبع هذا المجموع.

الحـــــــــــل:

لاحظ أن هذا البرنامج شبيه بالبرنامج السابق له، إلا أننا نريد طباعة مجموع الأعداد التي تقبل القسمة على ٧ في مدى معين، وليس طباعة قيم هذه الأعداد.

سنحتاج متغيراً لحفظ المجموع بشكل تراكمي، وليكن هذا المتغير هو sum، ولتكن قيمته في البداية صفراً، كوننا لم نبدأ بعملية المعالجة. ثم سنقوم، كما في البرنامج السابق، بإدخال النهاية الصغرى للأعداد، ولتكن lr والنهاية العظمى، ولتكن ur. ثم نقوم بقسمة النهاية الصغرى على ٧ ، إذا كان باقي القسمة صفر نضيف النهاية الصغرى إلى قيمة sum، والا نزيد النهاية الصغرى بمقدار ١، ثم نختبرها ، فإذا تجاوزت النهاية العظمى نتوقف، والا نعود إلى خطوة القسمة من جديد.

والشكل (١٧.٢) التالي يوضح خطوات الحل، وهي:

١- اجعل قيمة المتغير sum صفراً، أي: $sum = 0$.

٢- اقرأ النهاية الصغرى lr، والنهاية العظمى ur.

٣- أحسب باقي قسمة النهاية الصغرى على ٧ واحفظه في متغير r، كما في المعادلة: $r = lr \bmod 7$.

٤- إذا كانت قيمة باقي القسمة r تساوي صفراً، انتقل إلى الخطوة ٥، وإلا انتقل إلى الخطوة ٦:

٥- - أضف قيمة lr إلى sum، أي: $sum = sum + lr$.

- زد قيمة lr بمقدار ١، أي: $lr = lr + 1$. انتقل إلى الخطوة ٧.

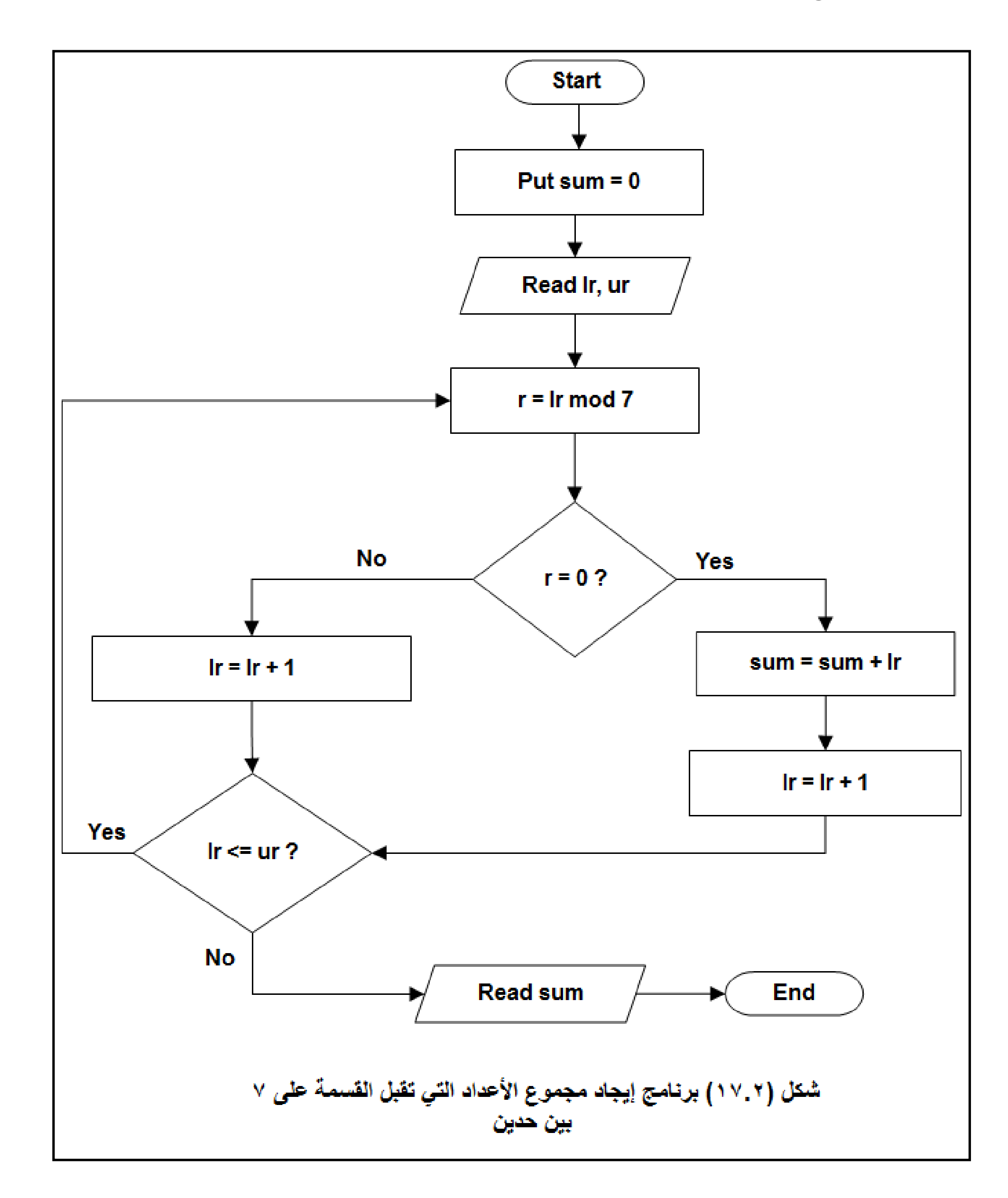
٦- زد قيمة lr بمقدار ١، أي: $lr = lr + 1$.

٧- - إذا كانت lr ما تزال أقل من أو تساوي ur، عد إلى بداية الخطوة ٣، وإلا انتقل إلى الخطوة التالية.

٨- توقف عن التكرار.

٩- أطبع قيمة المتغير sum.

١٠- انتهى البرنامج.



شكل (٢.١٧) برنامج إيجاد مجموع الأعداد التي تقبل القسمة على ٧ بين حدين

مثـــــــــــــال ٥.٣.٩.٢

ارسم خريطة سير البرنامج (flow chart) لبرنامج يقوم بتحديد نسبة المبيعات التي يحصل عليها مندوبو مبيعات يعملون في إحدى شركات المبيعات. وذلك تبعاً للقواعد التالية:

المبيعات	نسبة العمولة
< = 5000	٧% من نسبة المبيعات
> 5000 but < = 10000	٩% من نسبة المبيعات + ٥٠٠ ريال
> 10000 but < = 20000	١١% من نسبة المبيعات + ١٠٠٠ ريال
> 20000 but < = 25000	١٣% من نسبة المبيعات + ٢٠٠٠ ريال
> 25000	١٥% من نسبة المبيعات + ٤٠٠٠ ريال

(ويتوقف البرنامج إذا كانت كمية المبيعات المدخلة أقل من أو تساوي صفر).

الحـــــــــل:

في هذا البرنامج نقوم أولاً بإدخال مبلغ المبيعات sales، ثم نحسب العمولة commission حسب القواعد السابقة. والشكل (١٨.٢) يبين خطوات الحل، وهي كما يلي:

١- اقرأ مبلغ المبيعات التي باعها المندوب sales.

٢- قم بالخطوات التالية، طالما قيمة sales أكبر من الصفر، وعندما تصبح sales أقل من أو تساوي الصفر، انتقل إلى الخطوة ٤.

- إذا كانت قيمة sales أقل من أو تساوي ٥٠٠٠، احسب قيمة العمولة من المعادلة التالية:

$$commission = sales \times \frac{7}{100}$$

، ثم انتقل إلى الخطوة ٣.

- وإلا، إذا كانت قيمة sales أقل من أو تساوي ١٠٠٠٠، احسب قيمة العمولة من المعادلة التالية:

$$commission = sales \times \frac{9}{100} + 500$$

، ثم انتقل إلى الخطوة ٣.

- وإلا، إذا كانت قيمة sales أقل من أو تساوي ٢٠٠٠٠، احسب قيمة العمولة من المعادلة التالية:

$$commission = sales \times \frac{11}{100} + 1000$$

، ثم انتقل إلى الخطوة ٣.

- وإلا، إذا كانت قيمة sales أقل من أو تساوي ٢٥٠٠٠، احسب قيمة العمولة من المعادلة التالية:

$$commission = sales \times \frac{13}{100} + 2000$$

، ثم انتقل إلى الخطوة ٣.

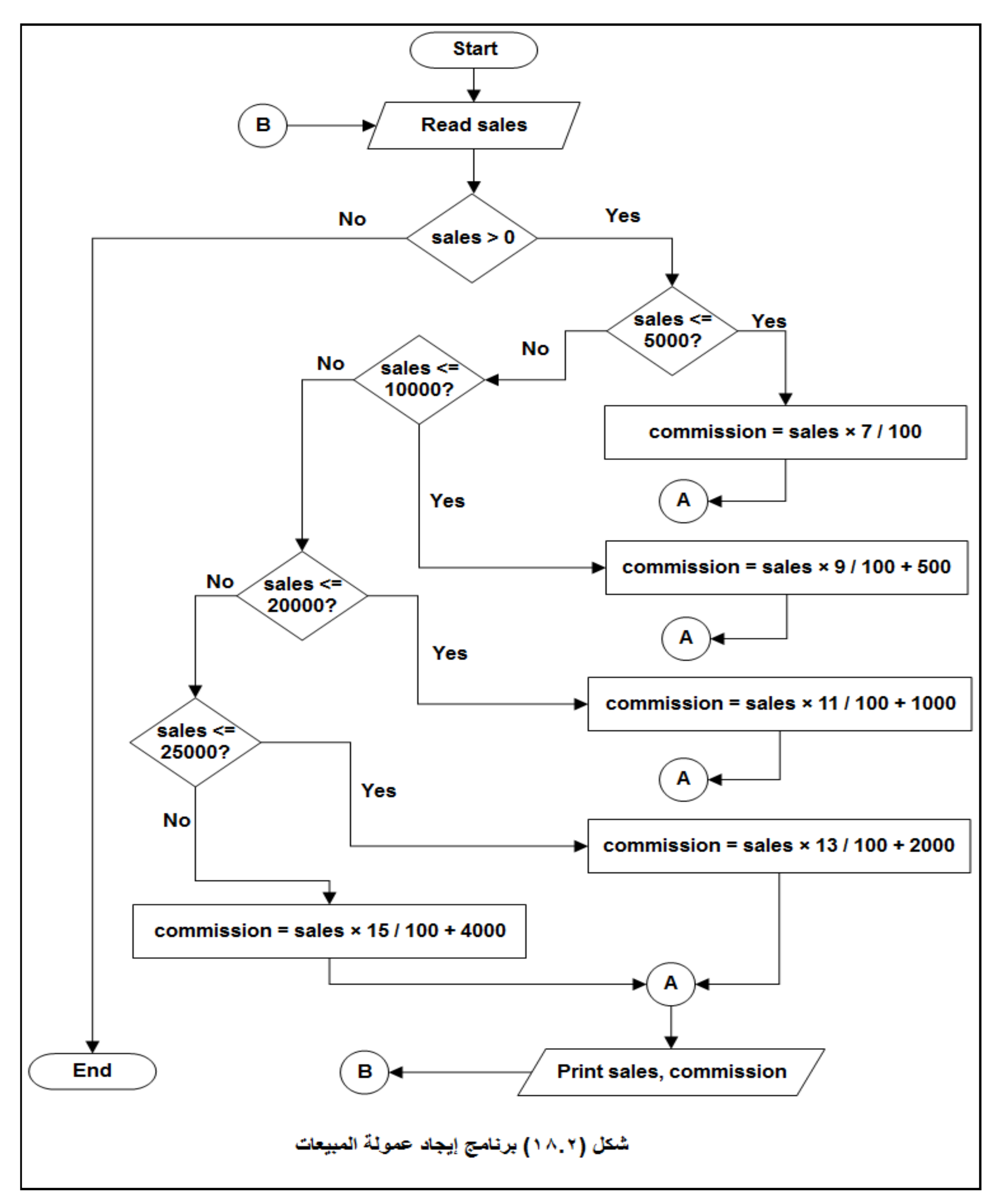
- وإلا، احسب قيمة العمولة من المعادلة التالية:

$$commission = sales \times \frac{15}{100} + 4000$$

٣- اطبع قيمتي sales، و commission، وارجع إلى الخطوة ١.

٤- توقف عن التكرار.

٥- انتهى البرنامج.



شكل (٢.١٨) برنامج إيجاد عمولة المبيعات

مثـــــــــــــــــال ٦.٣.٩.٢

ارسم خريطة سير البرنامج (flow chart) لبرنامج يقوم بطباعة خانات عدد طبيعي ما بشكل متفرق . مثلاً، إذا أدخلنا العدد ٣٩٥٦ يطبع ٦ ٥ ٩ ٣ .

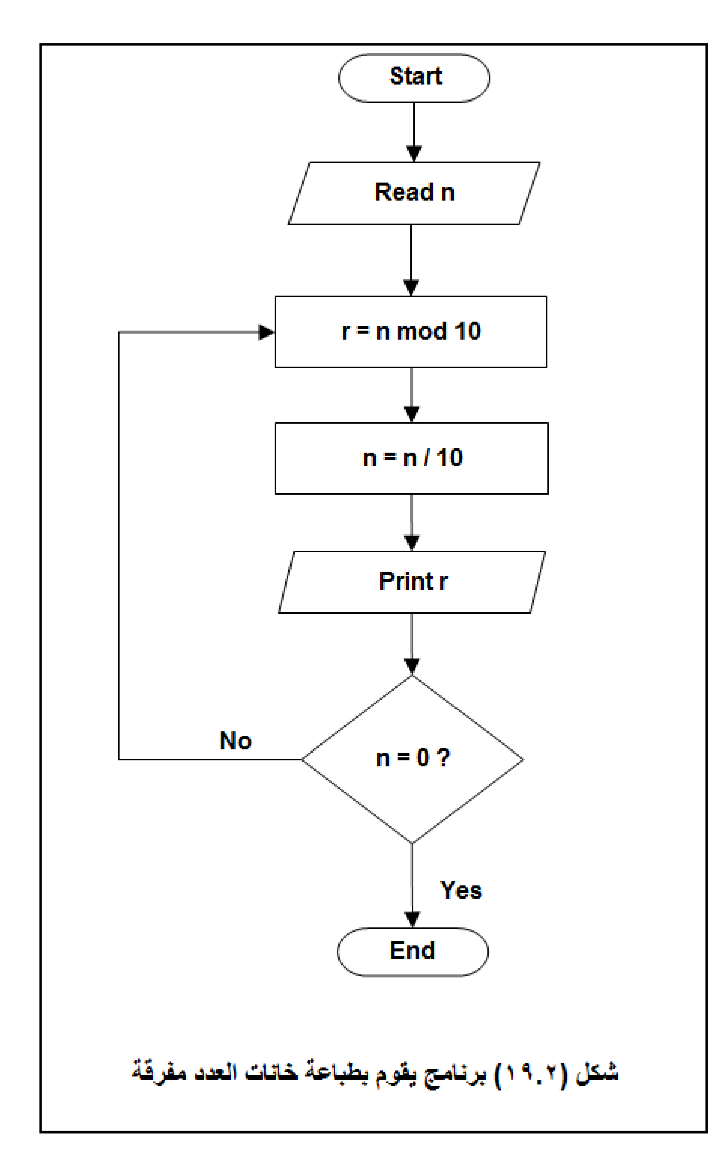
الحـــــــــــل:

واحدة من خوارزميات الحل لهذه المشكلة هي:

لنفرض أن العدد المدخل هو n . نقسم العدد n على ١٠ ونطبع باقي القسمة r في كل مرة ونطبع بعده مسافتان أو ثلاث. نستمر في هذه العملية حتى يصبح n مساوياً للصفر.

والشكل (١٩.٢) يبين خطوات الحل، والتي هي كما يلي:

١- اقرأ العدد n.
٢- احسب باقي قسمة n على ١٠، واحفظ الناتج في متغير وليكن r، أي: $r = n \bmod 10$.
٣- احسب ناتج القسمة الصحيح لـ n على ١٠، واحفظه في المتغير n نفسه، أي: $n = \frac{n}{10}$.
٤- اطبع قيمة r.
٥- إذا أصبحت n مساوية للصفر، انتقل إلى الخطوة التالية، والا عد إلى الخطوة ٢.
٦- توقف عن التكرار.
٧- انتهى البرنامج.



شكل (١٩.٢) برنامج يقوم بطباعة خانات العدد مفرقة